

ON PERMUTATIONS WITH DECIDABLE CYCLES

TOBIAS BOEGE

ABSTRACT. Recursive permutations whose cycles are the classes of a decidable equivalence relation are studied; the set of these permutations is called Perm , the group of all recursive permutations \mathcal{G} . Multiple equivalent computable representations of decidable equivalence relations are provided. \mathcal{G} -conjugacy in Perm is characterised by computable isomorphy of cycle equivalence relations. This result parallels the equivalence of cycle type equality and conjugacy in the full symmetric group of the natural numbers.

Conditions are presented for a permutation $f \in \mathcal{G}$ to be in Perm and for a decidable equivalence relation to appear as the cycle relation of a member of \mathcal{G} . In particular, two normal forms for the cycle structure of permutations are defined and it is shown that conjugacy to a permutation in the first normal form is equivalent to membership in Perm . Perm is further characterised as the set of maximal permutations in a family of preordered subsets of automorphism groups of decidable equivalences.

Conjugacy to a permutation in the second normal form corresponds to decidable cycles plus decidable cycle finiteness problem. Cycle decidability and cycle finiteness are both shown to have the maximal one-one degree of the Halting Problem. Cycle finiteness is used to prove that conjugacy in Perm cannot be decided and that it is impossible to compute cycle deciders for products of members of Perm and finitary permutations. It is also shown that Perm is not recursively enumerable and that it is not a group.

I. INTRODUCTION

An equivalence relation over the natural numbers is *decidable* if there is a Turing machine which decides for every pair (x, x') of numbers whether they are related or not. The set of all recursive permutations of the natural numbers is denoted \mathcal{G} in this paper. We consider the subset Perm in \mathcal{G} of recursive permutations whose orbits, or *cycles*, are the classes of a decidable equivalence relation. The present paper studies algorithmic and algebraic questions about Perm which arise naturally from this definition, which relates equivalence relations and permutations.

Identify a permutation f of \mathbb{N} with the digraph on vertices \mathbb{N} whose arrows are given by the mapping $x \rightarrow f(x)$. A permutation is in \mathcal{G} if its digraph is locally explorable by a Turing-computable algorithm. For the permutations in Perm a more global class of questions can be decided in addition, namely for any pair of numbers whether they belong to the same weakly connected component of the digraph. The weakly connected components in the digraph view correspond to the cycles of the permutation and we use these two terms synonymously.

The set Perm appears in an attempt to transfer a well-known theorem from Group Theory to Recursion Theory. This theorem states that in a symmetric group, such as $\text{Sym } \mathbb{N}$, conjugacy is equivalent to cycle type equality. Cycle type equality of two permutations is the condition that for each countable cardinal, the both permutations have the same number of cycles of size that cardinal. An attempt at this transfer has been made by Kent in 1962:

Theorem ([Ken62, Thm. 1.7]). In the group \mathcal{G} , a cycle type class is also a conjugacy class iff it is the cycle type class of a permutation with finitely many infinite cycles.

Define the set \mathcal{G}_1 to consist precisely of those recursive permutations with finitely many infinite cycles. Then Kent's theorem states that in \mathcal{G}_1 two permutations are \mathcal{G} -conjugate iff they have the same cycle type and there is no proper superset $\mathcal{G}_1 \subsetneq \mathcal{G}' \subseteq \mathcal{G}$ which is closed under cycle

Date: December 16, 2016.

2010 Mathematics Subject Classification. Primary: 03D45; Secondary: 20E45.

Key words and phrases. Permutation, equivalence relation, partition, conjugacy, cycle decidability, computability.

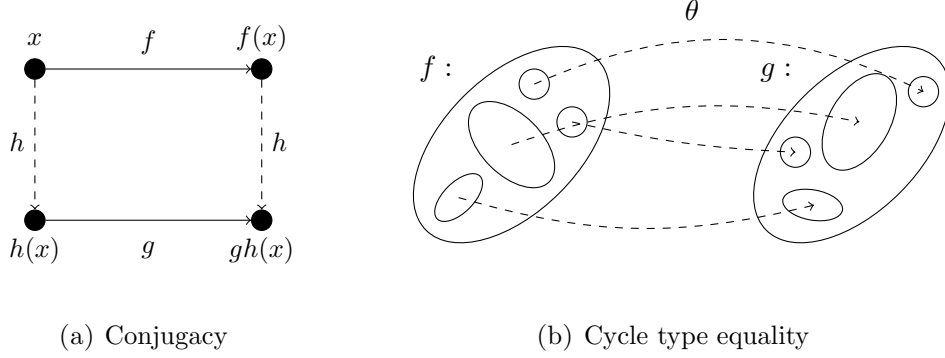


FIGURE 1. Witnesses of conjugacy and cycle type equality. Conjugacy of f and g is witnessed by an isomorphism h between their respective digraphs, cycle type equality by an isomorphism θ between the equivalence relations of *undirected* reachability in their respective digraphs. The solid circles in 1a are vertices, the ellipses in 1b are weakly connected components.

types and where this theorem holds, too. This formulation takes the same form as the one for full symmetric groups, but one might argue that \mathcal{G}_1 is a too small subset of \mathcal{G} . We will show, for instance, that cycle decidability and cycle finiteness are trivial problems in \mathcal{G}_1 (Propositions 4.2, 8.4). Thus one asks why the notions of conjugacy and cycle type equality drift apart in \mathcal{G} . The crucial observation is that by changing the group from $\text{Sym } \mathbb{N}$ to \mathcal{G} , the notion of conjugacy changes. Conjugation in \mathcal{G} is always afforded by a *recursive* permutation — it gets a constructive character. Cycle type equality, on the other hand, remains non-constructive in Kent’s theorem. If we define an effective version of cycle type equality, we are not restricted by the second part of Kent’s theorem anymore and may find a bigger set in which the new formulation of the theorem holds. To make cycle type equality effective, one first needs a witness for the condition of cycle type equality. Such a witness would be a bijection between the weakly connected components which preserves the size of each component, or, alternatively, a bijection between the sets of vertices which respects weakly connected components in both directions. If we denote the equivalence relation whose equivalence classes are the weakly connected components of f by \equiv_f , then we require a permutation θ of \mathbb{N} such that $x \equiv_f x' \Leftrightarrow \theta(x) \equiv_g \theta(x')$. This takes the form of an equivalence isomorphism; see Figure 1 for an illustration of witnesses for conjugacy and cycle type equality. The notions of equivalence relation and isomorphism thereof can easily be transferred into Recursion Theory and yield the desired definition of effective cycle type equality. The present paper starts by giving a number of possible definitions for decidable equivalences which parallel characterisations of equivalence relations in non-constructive Mathematics, and shows that they are equivalent, too, in Recursion Theory. Based on this solid notion of decidable equivalence, we are interested in recursive permutations f whose relation \equiv_f is decidable, as only for those permutations effective cycle type equality can be defined in the language of Recursion Theory. The set of these permutations is exactly Perm , and indeed one obtains an analogue to the theorem in $\text{Sym } \mathbb{N}$, our Theorem 3.8, which states that in Perm \mathcal{G} -conjugacy and effective cycle type equality are equivalent. This theorem has two advantages over the one in \mathcal{G}_1 : (1) the equivalence is constructive, i.e. a witness for either condition can be converted into one for the other by Turing-computable algorithms, and (2) \mathcal{G}_1 is properly (and trivially) contained in Perm , by Proposition 4.2.

After the theorem about the equivalence of \mathcal{G} -conjugacy and effective cycle type equality in Perm is established, section III gives a number of characterisations of Perm . The two main ideas come from the two ways to approach Perm , as per its definition. The first is the question whether a given $f \in \mathcal{G}$ has decidable cycles, the second is whether a given decidable equivalence relation is realisable as the cycle equivalence of a permutation in \mathcal{G} . In this process, §4 collects evident sufficient and equivalent extrinsic criteria for cycle decidability. The approach of §5 is

to characterise cycle decidability intrinsically through normal forms for the cycle structure of permutations. Two forms, the *normal* and the *semi-normal* form, are defined. A corollary to Theorem 5.8 is that a permutation has decidable cycles iff it is \mathcal{G} -conjugate to a permutation in normal form. Indeed if a permutation is in normal form, a decider for its cycles can be extracted from the permutation alone. Normality, in this light, is a uniformity condition. Then, §6 gives sufficient and equivalent criteria for a decidable equivalence to be *permutable*, i.e. to appear as the cycle equivalence of a recursive permutation. It is shown that decidable and permutable equivalences are in bijection with recursive normal-form permutations. The techniques developed in this subsection give powerful tools to construct permutations from mere equivalence relations which only encode the orbits of a permutation, not the cyclic structure. Especially Corollary 6.4 proves to be useful in existence theorems such as Proposition 6.5, Lemma 8.2 and Theorem 10.4. One further characterisation of Perm is of order-theoretic nature. It is shown that the elements of Perm are exactly the permutations in a preordered subset of the automorphism group of some decidable equivalence which are maximal with respect to cycle inclusion. Here, the characterisation via maximality does not use notions of computability; these occur only in the setting, i.e. the choice of the preordered sets.

While the normal form deals with cycle decidability, the semi-normal form specifies an incompatibility between the structure of finite and infinite cycles which makes it possible, in addition to deciding the cycles, to decide the *cycle finiteness problem* of the semi-normal permutation, that is for each number x to determine if its cycle is finite or infinite. It has previously been shown by Lehtonen [Leh09] that there are recursive permutations which can be defined by fairly elementary formulae but whose cycle finiteness problem is undecidable. In §8 in section IV, it is proved that the set of permutations for which this problem is decidable is precisely the union of \mathcal{G} -conjugacy classes of recursive semi-normal permutations. Remarkably, the sets of permutations for which the cycle decidability and cycle finiteness problems are solvable, each possess a set of generators with respect to \mathcal{G} -conjugacy, namely the normal and the semi-normal permutations, such that the problems are uniformly solvable on these generators.

The fact that cycle finiteness in general is undecidable in Perm can be applied in reductions. For example, deciding cycle finiteness is an instance of deciding conjugacy in Perm, which implies that the latter is also undecidable. As a corollary to a result by van Leeuwen [vL15], we obtain that Perm is not recursively enumerable. In §10 it is shown that cycle decidability problems in \mathcal{G} are at most as hard as the Halting Problem for Turing machines and that this upper bound is attained. It is also proved that the classes of cycle decidability problems and cycle finiteness problems in \mathcal{G} are reducible to each other. The last subsection concerns multiplicative closure of Perm. It is shown that a product of a member of Perm and a finitary permutation is again in Perm. In contrast to Theorem 3.8, this fact has no algorithmic content. The third part of Corollary 11.5 states that there is no computable mapping which associates a decider for the cycles of $af \in \text{Perm}$ to every triple (a, f, π) , where $a \in F$, F the set of finitary permutation, $f \in \text{Perm}$ and π a decider for the cycles of f . Such an algorithm exists, however, for semi-normal permutations. By Corollary 11.1, a product of two arbitrary (non-finitary) members of Perm does not necessarily reside in Perm which shows that Perm is not a group. This may be seen as an ultimate defect in trying to recover the aforementioned theorem on conjugacy in symmetric groups inside Recursion Theory. Theorem 3.8 is an entirely constructive version of this theorem, but its domain, Perm, is missing the group structure from the non-constructive original, which means that the notion of conjugacy has to be borrowed from the ambient group \mathcal{G} . Kent's set \mathcal{G}_1 suffered from the same problem. Indeed this failure is predetermined by a result on the composition series of \mathcal{G} , also due to Kent: no set strictly between F and \mathcal{G} which is closed under \mathcal{G} -conjugation, i.e. normal, can be a group.

If not explicitly introduced, the notation follows [Rog87], which can also serve as the primary resource for recursion-theoretic facts used without citation. The set of natural numbers, including 0, is denoted \mathbb{N} and the set of positive integers as \mathbb{N}^+ . We denote the domain of a partial function f by $\text{dom } f$ and its range by $\text{rng } f$. If a composition of functions is applied to an object, we save parentheses: instead of $f(g(x))$ or $(fg)(x)$ only the “application to x ”

parentheses are written: $fg(x)$. By $\langle x, y \rangle$ we denote a recursive *pairing function* which maps pairs (x, y) to numbers bijectively. Instead of $f(\langle x, y \rangle)$ we write $f\langle x, y \rangle$. Existential and universal quantifiers are taken over the natural numbers if no set is given. The complement of a set A is also understood as the complement in the natural numbers and written as \overline{A} . Usually we do not discriminate between a partial recursive function f and a program, or *Gödel number*, \mathbf{f} under the standard numbering of partial recursive functions: $\varphi_{\mathbf{f}} = f$. The associated standard numbering of recursively enumerable sets is $W_x := \text{dom } \varphi_x$. When the function f is used in the description of an algorithm, it is implied that any Gödel number for f would suffice. Most proofs in this paper are constructive and yield, by virtue of the Church-Turing Thesis, a (Turing-computable) algorithm. This is usually indicated by an *effectiveness* addition in the formulation of the statement. The term “uniformly effectively” is used throughout the text according to [Rog87, § 5.5]. The term “cycle” is used with two different meanings in this paper. Firstly, it can mean an orbit of the group action a permutation affords on its domain by function application. Secondly it abbreviates *cyclic permutation*, a permutation whose support consists of at most one orbit. Commonly f, g, h denote recursive permutations, often members of Perm , σ, ρ, ψ partial recursive functions encoding an object of importance in a limited scope, π, γ deciders for equivalences Π, Γ , and x, y, z natural numbers.

Acknowledgement. The author wishes to thank Thomas Kahle for his support of this paper and discussion of earlier versions which helped to improve the presentation.

II. DECIDABLE EQUIVALENCES AND THEIR ISOMORPHISMS

§1 Decidable equivalence relations. This subsection gives a number of possible constructive representations for decidable equivalence relations and shows that they all are effectively equivalent: for each pair of representations there is an algorithm which transforms one into the other.

A set $\Pi \subseteq \mathcal{P}(\mathbb{N})$ of subsets of \mathbb{N} is called a *partition* if the sets in Π are non-empty, pairwise disjoint and their union is \mathbb{N} . The elements of Π are called its *blocks*. Denote by $P(x, \Pi)$ the unique block $P \in \Pi$ with $x \in P$. Usually we abbreviate this to $P(x)$ if there is no danger of confusion. It is well known that the concepts of partition and equivalence relation are equivalent: to each partition Π there is the equivalence $x \equiv_{\Pi} x' :\Leftrightarrow P(x) = P(x')$ and the classes of every equivalence form a partition. We write $x \Pi x'$ instead of $x \equiv_{\Pi} x'$ and speak of Π also as an equivalence relation.

Definition 1.1. An equivalence relation Π over \mathbb{N} is *decidable*, if for every $x, x' \in \mathbb{N}$, the predicate $x \Pi x'$ is decidable.

Let \mathbf{t} and \mathbf{f} be two distinct natural numbers, representing *true* and *false*. We adopt the following syntax for predicates p :

$$[p(x_1, \dots, x_n)] := \begin{cases} \mathbf{t}, & p(x_1, \dots, x_n) \text{ is true,} \\ \mathbf{f}, & \text{else,} \end{cases}$$

Then we can state the decidability of Π as follows: there exists a recursive function π of two variables such that $\pi(x, x') = [x \Pi x']$.

For Recursion Theory, the need for representation of objects as natural numbers arises. A partition Π is identified with its set of blocks. By the symbol \equiv_{Π} we denote the same equivalence relation as Π but represented in the customary fashion as a set of ordered pairs $(x, x') \in \equiv_{\Pi} :\Leftrightarrow x \Pi x'$. Then we see that Π is a decidable equivalence iff \equiv_{Π} is a recursive set (of pairs) in the usual sense of [Rog87, § 5.3]. This subsection shows that the distinction of these two representations is immaterial to computability. The following lemma is obvious from the s_n^m Theorem:

Lemma 1.2. Π is decidable iff there is a computable function σ such that $\varphi_{\sigma(x)}$ decides $P(x)$ for all $x \in \mathbb{N}$. The decider π for Π and the function σ are computationally equivalent, in that either can be computed from the other. \square

If Π is decidable, then every block of Π is a decidable set. The converse is not true. As Lemma 1.2 suggests, a *uniform* decider for the blocks is needed. An example of an undecidable equivalence on \mathbb{N} whose blocks are all recursive can be manufactured from an undecidable problem in two variables which becomes decidable if one of its variables is fixed. Consider the two-variable decision problem, given a program x and a (suitable encoding of) a cardinal number $m \leq \aleph_0$ to decide if $|W_x| = m$. Write $C(x, m) = [|W_x| = m]$. By Rice's Theorem $C(x, m)$ is undecidable, however if x is fixed, it becomes trivial. Define an equivalence Π as

$$\langle x, m \rangle \Pi \langle x', m' \rangle : \Leftrightarrow x = x' \wedge C(x, m) = C(x, m').$$

The block $P(\langle x, m \rangle, \Pi) = \{\langle x, m' \rangle : C(x, m) = C(x, m')\}$ is a recursive set as $C(x, m)$ is computable for fixed x . If the equivalence as a whole was decidable, we could decide $E(x, m, m') = [C(x, m) = C(x, m')]$ uniformly in x, m, m' . Now let x, m be given. Choose two distinct cardinals p, q which are also distinct from m . At least two of $C(x, m), C(x, p), C(x, q)$ must be false. A complete discussion of the cases

$C(x, m)$	$C(x, p)$	$C(x, q)$	$E(x, m, p)$	$E(x, m, q)$	$E(x, p, q)$
f	f	t	t	f	f
f	t	f	f	t	f
t	f	f	f	f	t
f	f	f	t	t	t

characterises the statement $C(x, m) = \mathbf{t}$. Thus if we assume $E(x, m, m')$ to be uniformly computable in all its parameters, we can decide $|W_x| = m$ for any given x and m , which is a *contradiction*.

Proposition 1.3. Let A be a non-empty initial segment, i.e. either $A = \{0, \dots, n-1\}$ for some $n \in \mathbb{N}^+$ or $A = \mathbb{N}$. Then a partition $\Pi = \{P_i : i \in A\}$ is decidable iff there is a partial recursive function p such that if $i \in A$ it follows that $\varphi_{p(i)}$ decides P_i . A program for p can be obtained from a decider π for Π and vice versa.

Proof. “ \Rightarrow ”: Let π decide Π . It follows from Lemma 1.2 that every block is decidable. We need to enumerate a decider for every block of Π so that each block is represented exactly once among the first $|\Pi|$ values of p . Define $i(x) := \mu x' [x \Pi x']$ which is total recursive via π and returns the least representative of the block of x . Let σ be the function obtained from π by Lemma 1.2. Now

$$\begin{aligned} n(x) &:= \mu x' [\forall y < x : i(x') \neq in(y)], \\ p(x) &:= \sigma n(x) \end{aligned}$$

are both computable by appeal to the Church-Turing Thesis. p is obtainable uniformly effectively from π . We will prove that p is the wanted enumeration. The μ search in n for input x computes $n(y)$ for all $y < x$. It follows that if p halts on input x it must have halted on all inputs $y < x$ and if p does not halt on y it will not halt on inputs $x > y$. The set of inputs where p halts is an initial segment A of \mathbb{N} . If $x \in A$, $p(x) \in \text{rng } \sigma$ is a program to decide some block of Π . We claim that n possesses these three properties:

- (a) $in(x) = n(x)$,
- (b) $x < z \Rightarrow n(x) < n(z)$ and
- (c) $x \leq n(x)$

for all x, z . Clearly $in(x) \leq n(x)$ and furthermore $\forall y < x : in(x) = in(x) \neq in(y)$ where the inequality follows by definition of $n(x)$. This means that $in(x)$ also satisfies the condition over which $n(x)$ is minimised, so $n(x) \leq in(x)$ and it must hold equality. For the second property observe that the sets over which $n(x)$ and $n(z)$ minimise are isotone: $\{x' : \forall y < x : i(x') \neq$

$in(y)\} \subsetneq \{x' : \forall y < z : i(x') \neq in(y)\}$. The difference of these sets includes $n(x)$ so that indeed $n(x) < n(z)$. The last property follows by induction from the monotonicity: we have $0 = n(0)$ and if $x \leq n(x)$ for some x , it follows $x \leq n(x) < n(x+1)$ and thus $x+1 \leq n(x+1)$.

We can now prove that each block $P \in \Pi$ has at least one decider in $\text{rng } p$: Let j be the minimal element in P , then $i(j) = j$. If j was not already taken as an image $n(y)$ for $y < j$, it must hold that $\forall y < j : j \neq n(y)$. But $j = i(j)$ and $n(y) = in(y)$, so that $\forall y < j : i(j) \neq in(y)$, i.e. j satisfies the condition over which $n(j)$ minimises and $j \geq n(j)$. By the third property above also $n(j) \geq j$ and so $j = n(j)$ at the latest. On the other side, we see from the definition of n that $\forall y < x : in(x) \neq in(y)$, which means $n(x)$ and $n(y)$ are in different blocks. Therefore $\sigma n(x)$ decides a different block than $\sigma n(y)$ for $y < x$. Consequently each block has at most one decider. In conclusion p is partial recursive and halts precisely on an initial segment of \mathbb{N} . Wherever it halts it gives a program to decide a block of Π so that each block gets exactly one decider.

“ \Leftarrow ”: The function $j(x) := \mu j[x \in P_j]$ is computable by means of $p(j)$. It is a total function because Π is a partition. A is an initial segment of \mathbb{N} and every x will be found in some P_j for $j \in A$, i.e. before, in the μ search in $j(x)$, p must be evaluated for an argument $j \notin A$, where its behaviour is not specified. Then $x \Pi x' \Leftrightarrow j(x) = j(x')$. Hence $\pi(x, x') := [j(x) = j(x')]$ decides Π and can be obtained uniformly effectively from a program for p . \square

Another way to constructively define equivalence relations is to require a method which assigns each number a label. The equivalence classes are formed by grouping equally labeled numbers together:

Proposition 1.4. For each decidable equivalence Π there is a recursive function r whose non-empty fibers are the blocks of Π . Conversely the set of non-empty fibers of any recursive function is a decidable equivalence. There are algorithms to convert a decider for Π into a recursive function and vice versa.

Proof. If Π is decidable, the function $r(x) := \mu x'[x \Pi x']$ is computable. Obviously $r(x) = r(x') \Leftrightarrow x \Pi x'$. If r is any recursive function, then $\lambda x x'[r(x) = r(x')]$ is a recursive decider for the equivalence induced by the non-empty fibers of r . \square

The results thus far enable us to represent a decidable partition Π in one of four constructive ways: (1) as the recursive function π which decides the predicate $[x \Pi x']$, (2) as the σ function from Lemma 1.2, (3) as the partial recursive p from Proposition 1.3 which enumerates deciders for the blocks of Π , and (4) as the non-empty fibers of a recursive function. The function π decides \equiv_Π as a set of ordered pairs while p corresponds to the representation of the partition Π as a set of blocks. We have thus seen that these representations are computationally equivalent, i.e. we can go effectively from one representation to another. This result justifies our identification of partitions as sets of blocks with equivalence relations in the form of sets of ordered pairs.

We remark in passing that it is not critical that p enumerates deciders for the blocks; recursive enumerators are sufficient. The proof is via dove tiling.

Proposition 1.5. Let A be a non-empty initial segment and $\Pi = \{P_i : i \in A\}$. Then Π is decidable iff there is a partial recursive q such that $\text{dom } q = A$ and if $i \in A$, then $\varphi_{q(i)}$ enumerates P_i . A program for q can be obtained from a decider π for Π and vice versa. \square

§2 Decidable cycles. For any permutation f , the set of cycles of f form a partition of its domain. Denote the cycle of f to which x belongs by $[x]_f$. Let Φ temporarily denote the function which maps a permutation to its thus associated partition. Conversely, to any partition Π of a set, the inverse image $\Phi^{-1}(\Pi)$ is the set of permutations associated with it. For Recursion Theory, these considerations are restricted to decidable equivalences and recursive permutations.

Definition 2.1. If Π is a decidable equivalence, the set of associated permutations is $\text{Perm } \Pi := \Phi^{-1}(\Pi) = \{f \in \mathcal{G} : \forall x : P(x, \Pi) = [x]_f\}$. A recursive permutation f has an associated partition $\text{Part } f := \Phi(f) = \{[x]_f : x \in \mathbb{N}\}$. The corresponding equivalence relation, for infix usage, is denoted \equiv_f . The central object of study in this paper is the set

$$\text{Perm} := \bigcup_{\Pi \text{ decidable}} \text{Perm } \Pi.$$

Writing the cycle equivalence relation down explicitly, $x \equiv_f x' :\Leftrightarrow \exists k \in \mathbb{Z} : x' = f^k(x)$, it takes the form of a *reachability relation*. A recursive permutation f is in Perm precisely when reachability in its undirected functional graph is decidable.

It is in general not true that $\text{Part } f$ is a decidable partition for each f or that $\text{Perm } \Pi$ is non-empty for each decidable Π . Section III treats the questions of when these statements do hold extensively. At this point we give the relevant definitions only:

Definition 2.2. An equivalence Π is *permutable* if $\text{Perm } \Pi \neq \emptyset$. Conversely, a permutation f has *decidable cycles* if $\text{Part } f$ is a decidable equivalence.

In this and the following subsection we concentrate on the relation of the many possible permutations which belong to a single decidable partition Π and describe the conjugacy classes in Perm in terms of decidable equivalences.

Definition 2.3. In a subset of the group \mathcal{G} , two permutations f and g are \mathcal{G} -conjugate if there is $h \in \mathcal{G}$ such that $f = h^{-1}gh$. Instead of \mathcal{G} -conjugate we will use the term *effectively conjugate* or just *conjugate*. This relation is denoted as $f \sim g$.

A first step in understanding $\text{Perm } \Pi$ is provided by

Proposition 2.4. Let Π be decidable and $f, g \in \text{Perm } \Pi$. Then f and g are effectively conjugate. Moreover, given programs for f, g and a decider π for Π , a program to compute the conjugation between f and g can be described.

If f and g , as above, belong to the same equivalence relation, they have the same cycle type and would be conjugate in the full (non-constructive) symmetric group of the natural numbers. Although \mathcal{G} is not a symmetric group, the standard proof, as found in [BMMN98, Thm. 2.9] carries through under the additional premises of the proposition. A major non-constructive step in their proof is to pair cycles from f and g with matching lengths. This problem does not arise in our scenario because for each x we know that the cycle in f to which x belongs has the same length as the cycle for x in g , since f and g are permutations associated to the same partition. It is also convenient that the cycles for x in f and in g contain the same elements. Before we give the proof, we introduce the ξ operator which is the μ operator equivalent for (signed) integers. Its introduction is motivated by the observation that powers of permutations act like integers modulo cycle length.

Definition 2.5.

- (a) Let $\delta : \mathbb{Z} \rightarrow \mathbb{N}$ denote the (intuitively computable) bijection

$$\delta(k) := \begin{cases} 0, & k = 0, \\ -2k - 1, & k < 0, \\ 2k, & k > 0. \end{cases}$$

- (b) A partial function $\beta : \mathbb{Z} \rightsquigarrow \mathbb{Z}$ is called *partial recursive* if $\delta\beta\delta^{-1} : \mathbb{N} \rightsquigarrow \mathbb{N}$ is a partial recursive function.
- (c) Let $p : \mathbb{Z} \rightsquigarrow \mathbb{Z}$ be a partial recursive function. By writing $\xi k[p(k)]$ we mean that k is an integer and we perform *alternating search*, that is we evaluate $p(0), p(-1), p(1), p(-2), p(2), \dots$ in this order, until p becomes \mathfrak{t} for the first time. The first argument k in this process achieving $p(k) = \mathfrak{t}$ becomes the value of the ξ expression and if no such argument exists or p is undefined on some argument on the way, ξ does not terminate.

Lemma 2.6. $p : \mathbb{Z} \rightsquigarrow \mathbb{Z}$ be a partial recursive function. Then $\xi k[p(k)]$ is partial recursive.

Proof. Observe that $\xi k[p(k)] = \delta^{-1} \mu i[\delta p \delta^{-1}(i)]$ with the exact same semantics. \square

The δ function takes an integer k and, to produce the output natural number, doubles the absolute value of k and subtracts from this the sign bit of k . To compute the inverse of δ , examine the parity of the input, as it determines the sign of the output. The absolute value of the output integer is half of the input, rounded up to the next integer. Based on these intuitions for dealing with δ and δ^{-1} , and by appeal to the Church-Turing Thesis, the function $\lambda x x' \xi k[f^k(x) = x']$ is partial recursive and total on pairs (x, x') with $x \equiv_f x'$.

Proof of Proposition 2.4. We construct the conjugation h . Let π decide Π and x be given, then calculate $y(x) := \mu y[x \Pi y]$ via π . This calculation terminates as $x \Pi x$, so $y(x) \leq x$. By alternating search we determine $k(x) := \xi k[f^k y(x) = x] \in \mathbb{Z}$ which must exist. Return $h(x) := g^{k(x)} y(x)$. The following facts are immediate from the definition of h and the prerequisites:

- (i) $y(z) = y(x)$ iff $z \in [x]_f$ iff $z \Pi x$ iff $z \in [x]_g$,
- (ii) for any x the cycle lengths $|[x]_f|$ and $|[x]_g|$ match by (i),
- (iii) for any x the value $y(x)$ is a fixed point of h ,
- (iv) $f^k y(x) = f^{k'} y(x)$ iff $g^k y(x) = g^{k'} y(x)$ by (ii),
- (v) if $z \Pi x$ then $h(z) = g^{k(z)} y(x)$ by (i).

Because $f^{k(x)} y(x) = x$, it follows that

$$h f^{k(x)} y(x) = g^{k(x)} y(x)$$

holds for all inputs x . We want to modify this equation so that the exponent k is free in \mathbb{Z} . Take any $x \in \mathbb{N}$ and $k \in \mathbb{Z}$ and set $z = f^k y(x)$ so that $z \Pi x$. There is also the representation $z = f^{k(z)} y(x)$ by (v). This yields $h(z) = g^{k(z)} y(x) = g^k y(x)$ by (iv), which implies

$$(*) \quad h f^k y(x) = g^k y(x)$$

for all $x \in \mathbb{N}$ and $k \in \mathbb{Z}$.

To see that h is a permutation, assume first $h(x) = h(x')$, i.e. $g^{k(x)} y(x) = g^{k(x')} y(x')$. Because $y(x) = g^{k(x')-k(x)} y(x')$, $y(x)$ and $y(x')$ are in the same cycle of g and hence in the same cycle of f and must therefore be equal, as values in the range of y . We have shown $g^{k(x)} y(x) = g^{k(x')} y(x)$ which implies $x = f^{k(x)} y(x) = f^{k(x')} y(x) = x'$. To show surjectivity, let z be given. There is a representation as $z = g^k y(z)$ for some $k \in \mathbb{Z}$ because $z \in [y(z)]_f = [y(z)]_g$. But then already $h f^k y(z) = g^k y(z) = z$ by (*). This shows that h is a permutation, and for all $x = f^k y(x)$ we see via (*) that

$$h^{-1} g h(x) = h^{-1} g h f^k y(x) = h^{-1} g^{k+1} y(x) = f^{k+1} y(x) = f(x),$$

which completes the proof. \square

§3 Isomorphism and conjugacy. Proposition 2.4 shows that two permutations with identical equivalences are conjugate. The converse is not true as the following simple example shows: let $f = (\dots 6 \ 2 \ 0 \ 4 \ 8 \ \dots)$ and $g = (\dots 7 \ 3 \ 1 \ 5 \ 9 \ \dots)$. They define different equivalences but are conjugate via $(0 \ 1)(2 \ 3)(4 \ 5) \dots$. However, the equivalences defined by f and g are “essentially the same”, in that they have the same block structure. This observation suggests that studying equivalence isomorphisms leads to a better understanding of effective conjugacy classes.

Definition 3.1. Π and Γ be equivalences. A recursive permutation θ is a (Π, Γ) -isomorphism if $x \Pi y \Leftrightarrow \theta(x) \Gamma \theta(y)$. Isomorphy of Π and Γ is written as $\Pi \cong \Gamma$.

The next goal is a characterisation for the solvability of conjugation equations $f = h^{-1} g h$ when f, g have decidable cycles, and an algorithm for computing the solution h if a witness for the solvability is available. This result extends Proposition 2.4. To this end, some preparations are needed.

Lemma 3.2. Let Π be a decidable equivalence, h a recursive permutation. Then $\Pi^h := h(\Pi) := \{h(P) : P \in \Pi\}$ is a decidable partition.

Proof. Since h is a permutation, Π^h is again a partition. Then $x\Pi^h x'$ iff $P(x, \Pi^h) = P(x', \Pi^h)$, which is the case iff x and x' belong to the same $h(P)$, for some $P \in \Pi$. But this is equivalent to $h^{-1}(x)$ and $h^{-1}(x')$ belonging to the same $P \in \Pi$, i.e. $h^{-1}(x)\Pi h^{-1}(x')$. Thus $\lambda x x' [h^{-1}(x)\Pi h^{-1}(x')]$ is a recursive decider for Π^h . \square

Lemma 3.3. Let f be any recursive permutation. Then $x\Pi x' \Leftrightarrow f(x)\Pi^f f(x')$.

Proof. Π^f is a decidable partition. Then clearly $x\Pi x' \Rightarrow f(x)\Pi^f f(x')$. The converse follows by repeating the argument with f^{-1} in place of f . \square

Proposition 3.4. If θ is a recursive permutation, then it is a (Π, Γ) -isomorphism iff $\Gamma = \Pi^\theta$.

Proof. “ \Rightarrow ”: If θ is an isomorphism, we have $y\Gamma y' \Leftrightarrow \theta^{-1}(y)\Pi\theta^{-1}(y') \Leftrightarrow y\Pi^\theta y'$ according to Lemma 3.3. Thus $\Gamma = \Pi^\theta$.

“ \Leftarrow ”: $x\Pi x' \Leftrightarrow \theta(x)\Pi^\theta\theta(x') \Leftrightarrow \theta(x)\Gamma\theta(x')$. \square

The set of (Π, Π) -isomorphisms is the *automorphism group* of Π , denoted $\text{Aut } \Pi$. This is indeed a group under composition (a subgroup of \mathcal{G}) as $x\Pi x' \Leftrightarrow \text{id}(x)\Pi \text{id}(x')$, hence $\text{id} \in \text{Aut } \Pi$. If $f, g \in \text{Aut } \Pi$, then by Proposition 3.4: $\Pi^{fg} = (\Pi^g)^f = \Pi^f = \Pi$ and $\Pi^{f^{-1}} = (\Pi^f)^{f^{-1}} = \Pi^{f^{-1}f} = \Pi$ so that $fg, f^{-1} \in \text{Aut } \Pi$.

Lemma 3.5. Let Π be decidable and $f \in \mathcal{G}$. Consider the predicate

$$(*) \quad \forall x, x' : x\Pi x' \Leftrightarrow f(x)\Pi x'.$$

Then $f \in \text{Perm } \Pi \Rightarrow (*) \Rightarrow f \in \text{Aut } \Pi$. In particular $\text{Perm } \Pi \subseteq \text{Aut } \Pi$.

Proof. If $f \in \text{Perm } \Pi$, it follows that $x\Pi f(x)$. Together with $x'\Pi x$, this implies $x'\Pi f(x)$. With $f \in \text{Perm } \Pi$ is also $f^{-1} \in \text{Perm } \Pi$ and thus $f(x)\Pi x' \Rightarrow x = f^{-1}f(x)\Pi x'$.

Now assume $f \in \mathcal{G}$ and $(*)$ holds. By symmetry of Π and twofold application of $(*)$, we obtain $x\Pi x' \Leftrightarrow f(x)\Pi f(x')$, i.e. $\Pi = \Pi^f$ by Lemma 3.3. Then it follows that $f \in \text{Aut } \Pi$ by Proposition 3.4. \square

With Lemma 3.5 we are in a position to approach the following problem: given $f \in \text{Perm } \Pi$, Π decidable, and h a recursive permutation, a member of $\text{Perm } \Pi^h$ is to be described. If we assume that there is some $g \in \text{Perm } \Pi^h$, it would satisfy the relation $g(y)\Pi^h y' \Leftrightarrow y\Pi^h y' \Leftrightarrow h^{-1}(y)\Pi h^{-1}(y') \Leftrightarrow f h^{-1}(y)\Pi h^{-1}(y') \Leftrightarrow h f h^{-1}(y)\Pi^h y'$, by Lemmata 3.5 and 3.3. This shows that, at least under the assumption that Π^h is permutable, $h f h^{-1} \in \text{Aut } \Pi^h$. The following proposition strengthens this deduction in two ways: indeed $h f h^{-1} \in \text{Perm } \Pi^h$ and this is independent of the assumption that Π^h is permutable. A technical lemma is needed, which is obvious from the definition of $\text{Perm } \Pi$:

Lemma 3.6. $f \in \text{Perm } \Pi$ iff $f \in \mathcal{G}$ and for all x_0 the function $\mathbb{Z} \ni k \mapsto f^k(x_0)$ is surjective on $P(x_0, \Pi)$. \square

Proposition 3.7. Let $f \in \text{Perm } \Pi$ and $h \in \mathcal{G}$, then $h f h^{-1} \in \text{Perm } \Pi^h$.

Proof. Let $y_0 = h(x_0)$ and $y = h(x) \in P(y_0, \Pi^h)$ be given. Then $y_0\Pi^h y$ implies $x_0\Pi x$ by Lemma 3.3. Because $f \in \text{Perm } \Pi$, we can find k so that $f^k(x_0) = x$. Now $(h f h^{-1})^k(y_0) = h f^k h^{-1}(y_0) = h f^k(x_0) = h(x) = y$. It remains to show that $h f^k h^{-1}(y_0)$ is in $P(y_0, \Pi^h)$ for every k . Because $f \in \text{Perm } \Pi$, it is $f^k(x_0) \in P(x_0, \Pi)$ and thus $h f^k(x_0) \in P(y_0, \Pi^h)$ by definition of Π^h . Lemma 3.6 shows that $h f h^{-1} \in \text{Perm } \Pi^h$. \square

We remark that a more satisfying proof of this proposition can be given with the notions introduced in §7. It is not difficult to prove that the condition $(*)$ is characteristic for the set $\mathcal{I}\Pi$ which lies between $\text{Perm } \Pi$ and $\text{Aut } \Pi$ and is equipped with a preorder in §7. The mapping $f \mapsto h f h^{-1}$ is an order isomorphism $\mathcal{I}\Pi \rightarrow \mathcal{I}\Pi^h$ and it is shown that $\text{Perm } \Pi$ is the set of upper

bounds on $\mathcal{I}\Pi$. Since an order isomorphism preserves upper bounds, it follows that hfh^{-1} is an upper bound on $\mathcal{I}\Pi^h$ which immediately yields $hfh^{-1} \in \text{Perm } \Pi^h$.

We have remarked before the proof of Proposition 2.4 that conveniently the permutations f and g for a decidable partition Π have not only the same cycle type and that the cycles are paired already according to their length but the cycles of f and g to which an x belongs contain the same elements. Replacing identity of cycles with isomorphy yields a characterisation of conjugacy for members of Perm :

Theorem 3.8. Let f, g be recursive permutations for decidable partitions Π, Γ respectively. Then f and g are effectively conjugate iff Π and Γ are isomorphic. Put another way:

$$\forall f, g \in \text{Perm} : f \sim g \Leftrightarrow \text{Part } f \cong \text{Part } g.$$

If a decider for one of the equivalences is known, a conjugation between f and g can be obtained uniformly effectively from an isomorphism of the equivalences and vice versa.

Proof. “ \Rightarrow ”: Let $f = h^{-1}gh$ for some recursive permutation h . We claim that h is a (Π, Γ) -isomorphism. By Proposition 3.4 this is equivalent to the statement $\Pi^h = \Gamma$. By Proposition 3.7 we have $f = h^{-1}gh \in \text{Perm } \Gamma^{h^{-1}}$, but also $f \in \text{Perm } \Pi$. Thus the blocks of Π are the cycles of f which are also the blocks of $\Gamma^{h^{-1}}$. This implies $\Pi = \Gamma^{h^{-1}}$ and since h is a permutation, $\Pi^h = \Gamma$.

“ \Leftarrow ”: Let θ be a (Π, Γ) -isomorphism. Then $\theta^{-1}g\theta$ is in $\text{Perm } \Pi$ because by Proposition 3.7: $\theta^{-1}g\theta \in \text{Perm } \Gamma^{\theta^{-1}}$ and $\Gamma^{\theta^{-1}} = \Pi$ by Proposition 3.4. By virtue of Proposition 2.4 there is a recursive permutation h such that $f = h^{-1}\theta^{-1}g\theta h$, thus f and g are effectively conjugate via θh . \square

In view of permutations $f, g \in \text{Perm}$, the isomorphy of their respective equivalences $\text{Part } f \cong \text{Part } g$ is also called *effective cycle type equality*. From Theorem 3.8 it follows that if f has decidable cycles and Π is decidable and permutable, then

$$[f]_{\sim} = \bigcup_{\Gamma \cong \text{Part } f} \text{Perm } \Gamma,$$

$$[\Pi]_{\cong} = \{\text{Part } g : \exists f \in \text{Perm } \Pi : g \sim f\},$$

and using the former equation

$$\text{Perm} = \bigcup_{\Pi \text{ dec.}} \text{Perm } \Pi = \bigcup_{\text{Part } f \text{ dec.}} [f]_{\sim}.$$

Since Perm is a union of conjugacy classes, we obtain

Corollary 3.9. Perm is a normal subset of the group of recursive permutations. \square

III. CHARACTERISATIONS OF Perm

The definition of Perm establishes a relation between the permutations of \mathbb{N} , $\text{Sym } \mathbb{N}$, and the equivalence relations over \mathbb{N} , $\text{Eqv } \mathbb{N}$. Regarding the properties of “recursiveness” and “decidability”, this relation is non-trivial, as each of the four cases in Figure 2 is possible. From this relation arise two approaches to characterise Perm . The first starts in the “recursive permutation” column and asks when the permutation has decidable cycles and the second starts in the “decidable equivalence” row asks when the equivalence is permutable. The global questions whether every recursive permutation has decidable cycles and whether all decidable equivalences are permutable are both answered in the negative, as referenced in Figure 2. This section follows these two directions and also gives an order-theoretic characterisation of Perm .

§4 Cycle decidability. This subsection gives cycle decidability criteria, i.e. how must a recursive permutation be designed so that its cycles induce a decidable equivalence. There is a sufficient condition which is practically useful because it only requires some information from the cycle type of f :

Sym \mathbb{N} \diagdown Eqv \mathbb{N}	recursive	non-recursive
	Perm	Proposition 6.5
decidable	Perm	Proposition 6.5
undecidable	Lemma 10.1	$\Pi = \{K, \overline{K}\}$

FIGURE 2. All possibilities for the relation of recursiveness of permutations and decidability of equivalences are realisable. The equivalence $\Pi = \{K, \overline{K}\}$, with K the Halting Problem, is undecidable because its blocks are undecidable. Any permutation which has \overline{K} as a cycle is not recursive because the cycles of a recursive permutation are recursively enumerable.

Lemma 4.1. If f is a recursive permutation with at most one infinite cycle, then $\Pi = \text{Part } f$ is decidable. A decider for Π can be computed from f .

Proof. Let x, x' be given. If $x = x'$, the case is clear and we report $\pi(x, x') := \mathbf{t}$. In the other case, search

$$i = \mu i[i \geq 1 \wedge \{f^i(x), f^i(x')\} \cap \{x, x'\} \neq \emptyset].$$

This search always terminates:

- (i) If both x and x' are in an infinite cycle, they must be in the same cycle. Because $x \neq x'$ at this point, there is an $i \geq 1$ such that $f^i(x) = x'$ or $f^i(x') = x$, whereas $f^i(x) = x$ or $f^i(x') = x'$ is impossible for $i \geq 1$.
- (ii) If they are in the same finite cycle, we will find $f^i(x) = x'$ or $f^i(x') = x$ before $f^i(x) = x$ or $f^i(x') = x'$ can be encountered.
- (iii) If they are in different cycles $f^i(x) = x' \vee f^i(x') = x$ is unsatisfiable. But at least one of them must be in a finite cycle and thus $f^i(x) = x$ or $f^i(x') = x'$ will be found.

These are all the cases. If $f^i(x) = x'$ or $f^i(x') = x$, report $\pi(x, x') := \mathbf{t}$ and else $\pi(x, x') := \mathbf{f}$. The function π decides the cycles of f and can be computed uniformly effectively from f . \square

Lemma 4.1 can be generalised to the case of finitely many infinite cycles, but the constructed decider is not uniform in f anymore. The construction requires a system of representatives for the infinite cycles. For this reason, it is given as a separate proposition. This result can already be found in Myhill's paper [Myh59, Cor. 1] on *splinters* in digraphs of recursive functions, which generalise cycles in permutations. Myhill's corollary implies that all cycles of a permutation are recursive sets if there are only finitely many infinite cycles. As seen in §1, recursivity of all blocks of a partition does not suffice for (uniform) decidability of the partition. That addition, however, follows easily from his Theorem 1.3 and our characterisation of cycle decidability via transversals in Theorem 4.4 below. We give an independent proof here:

Proposition 4.2. If f is a recursive permutation with finitely many infinite cycles, then $\Pi = \text{Part } f$ is decidable. Using the definition from the introduction: $\mathcal{G}_1 \subseteq \text{Perm}$.

Proof. Let x_1, \dots, x_n be a *system of representatives* for the infinite cycles, i.e. each x_i belongs to an infinite cycle, no two of them belong to the same, and every infinite cycle has a representative among the x_i 's. Given x, x' we calculate indices k, k' respectively. If $x = x'$, set $k = k' = 0$. If $x = x_i$, then set $k = 0$, and analogously for x' and k' . Otherwise k is determined by alternating search of $f^k(x)$ in the set $\{x, x', x_1, \dots, x_n\}$: $k = \xi k[k \neq 0 \wedge f^k(x) \in \{x, x', x_1, \dots, x_n\}]$ and k' analogously by searching for $f^{k'}(x')$ in the same set. This procedure terminates because x is either in a finite cycle (and will find itself eventually with an index $k \neq 0$) or in an infinite cycle (and will find one of the representatives for infinite cycles with an index $k \neq 0$).

Once k, k' are computed, let $y = f^k(x), y' = f^{k'}(x')$ be the found elements. Note that $y, y' \in \{x, x', x_1, \dots, x_n\}$. If $y = x'$ or $y' = x$, then x and x' are in the same cycle. If $y = x \neq x'$ or $y' = x' \neq x$, then one of x, x' is in a finite cycle different from the other's. Otherwise $y = x_i$ and $y' = x_j$ for $1 \leq i, j \leq n$. Then $x \Pi x' \Leftrightarrow x_i = x_j$. This gives a way to decide Π . \square

Kent's [Ken62, Thm. 1.7], as quoted in the introduction, states that for the permutations with finitely many infinite cycles, cycle type equality and effective conjugacy are equivalent. By Theorem 3.8, in turn, effective conjugacy in Perm and effective cycle type equality are equivalent. The preceding Proposition 4.2 yields that \mathcal{G}_1 is contained in Perm and it follows

Corollary 4.3. In \mathcal{G}_1 cycle type equality and effective cycle type equality are the same. \square

For characterisations of cycle decidability, in contrast to Lemma 4.1 and Proposition 4.2, much more detailed information about the cycles is required.

Theorem 4.4. Let f be a recursive permutation. The following conditions are equivalent:

- (a) $\exists \pi : \pi(x, x') = [x \equiv_f x']$ (cycle decidability)
- (b) $\exists \mu : \mu(x) = \mu x' [x \equiv_f x']$ (smallest cycle representative)
- (c) $\exists \vartheta : (x \equiv_f x' \Rightarrow \vartheta(x) = \vartheta(x')) \wedge \vartheta(x) \equiv_f x$ (unique cycle representative)
- (d) $\exists \chi : \chi(x) = \chi(x') \Leftrightarrow x \equiv_f x'$ (characteristic value of a cycle)
- (e) $\exists \rho : (\rho(e) \equiv_f \rho(e') \Rightarrow \rho(e) = \rho(e')) \wedge \forall x \exists e : x \equiv_f \rho(e)$ (transversal)

where all functions are taken to be total recursive. These functions are pairwise computationally equivalent.

The condition (e) describes a recursively enumerable system of representatives for all cycles. Such a system is called a *choice set* by [Ken62] and [Myh59], and a *transversal* by [Hig90]. We adopt the term “transversal”. Higman furthermore calls the transversal consisting of the smallest elements of each cycle *principal*, which corresponds to condition (b) above. As the theorem shows, any recursively enumerable transversal of a permutation is computationally equivalent to the principal transversal.

Proof. We will show (a) \Rightarrow (b) \Rightarrow (e) \Rightarrow (a) and (b) \Rightarrow (c) \Rightarrow (d) \Rightarrow (b).

“(a) \Rightarrow (b)”: $\mu(x) := \mu x' [\pi(x, x')]$ is computable.

“(b) \Rightarrow (e)”: We need to enumerate representatives of all the cycles without representing a cycle twice with different values (it is allowed to repeat values because f may have only finitely many cycles). This can be done by $\rho = \mu$.

“(e) \Rightarrow (a)”: Given x, x' , start dove-tailing computations of $f^k \rho(e)$ where e varies in \mathbb{N} and k varies in \mathbb{Z} . Since ρ enumerates a complete system of representatives we will eventually find e, e' and k, k' such that $f^k \rho(e) = x$ and $f^{k'} \rho(e') = x'$. Because cycles are uniquely represented in ρ , it holds $x \equiv_f x' \Leftrightarrow e = e'$.

“(b) \Rightarrow (c)”: $\vartheta = \mu$ is possible.

“(c) \Rightarrow (d)”: $\chi = \vartheta$ is possible because $x \equiv_f x' \Rightarrow \vartheta(x) = \vartheta(x')$, and since $x \equiv_f \vartheta(x)$, we have $x \not\equiv_f x' \Rightarrow \vartheta(x) \neq \vartheta(x')$.

“(d) \Rightarrow (b)”: Take $\mu(x) := \mu x' [\chi(x) = \chi(x')]$. \square

[Myh59, Thm. 1.4] shows in the more general context of digraphs of recursive functions (not necessarily permutations), but with the same technique as above, that a recursively enumerable transversal for all weakly connected components, which are the corresponding generalisation of cycles, implies decidability of every component.

Aside from the transversal criterion, all characterisations of cycle decidability listed in Theorem 4.4 come in the form of an external function which answers questions about the cycles of the permutation. The next subsection follows a more profound approach to cycle decidability. We define a normal form for permutations, an intrinsic property of the permutation, and prove that conjugacy to a normal form permutation is equivalent to cycle decidability.

§5 Normal forms. Let f be a recursive permutation. Then for every block $P \in \text{Part } f$ and arbitrary $x_P \in P$, the function $\lambda j [f^{\delta^{-1}(j)}(x_P)]$ enumerates P . Thus $\text{Part } f$ is a partition whose blocks are recursively enumerable by the powers of a single fixed permutation at appropriately chosen starting points. Proposition 1.3 shows that if $\text{Part } f$ is decidable, every block of it must be decidable. It is well-known that a set is recursive iff it is recursively enumerable in non-decreasing order. We use this idea to define a normal form for permutations with decidable

cycles and derive a characterisation for permutable decidable equivalences. The following lemma provides an important technique for constructing a cycle out of an infinite recursive set. Some variants of the idea will also be used later.

Lemma 5.1. Let p be the characteristic function of an infinite recursive set A . Then we can find a cyclic permutation whose support is A .

Proof. Let x be given. $[x \in A]$ is computable by p . If $x \notin A$, then return $f(x) := x$. Else list the members of A in increasing order:

$$\begin{aligned} a(0) &:= \mu x[x \in A], \\ a(n+1) &:= \mu x[x > a(n) \wedge x \in A] \end{aligned}$$

which is uniform in p and total recursive because A is infinite. Then define

$$f(x) := a\delta(\delta^{-1}a^{-1}(x) + 1), x \in A.$$

The δ function is indeed used here according to Definition 2.5: Denote by succ the successor function $\text{succ}(x) = x + 1$ which is a recursive permutation on \mathbb{Z} . Then $f = a\delta \text{succ} \delta^{-1}a^{-1}$ where $\delta \text{succ} \delta^{-1}$ is a recursive permutation $\mathbb{N} \rightarrow \mathbb{N}$. This definition of f takes members of A into A injectively, as it is a composition of injective functions. For surjectivity it suffices to observe that a is surjective on A , and that $\delta \text{succ} \delta^{-1}a^{-1}$ maps A onto \mathbb{N} . It is clear that f has no fixed points in A . \square

In the normal form for cyclic permutations the structure of the cycle (repeated application of the permutation or its inverse) shall exhibit the increasingly ordered list of all members of the cycle, in an algorithmically recognisable way. A first attempt would be to take all members of the cycle in increasing order, $a_0 < a_1 < a_2 < \dots$ and define the cycle to be $(a_0 \ a_1 \ a_2 \ \dots)$. This layout is easy to work with but yields a permutation only if the number of a_i 's is finite, for if not a_0 will not get an inverse image. For infinite cycles, the idea of using δ from the proof of Lemma 5.1 comes into play:

Definition 5.2. A (finite or infinite) cycle $f = (\dots \ a_{-1} \ a_0 \ a_1 \ \dots)$ of length $n \in \mathbb{N}^+ \cup \{\infty\}$ whose least element is a_0 is in *normal form* if the function $\lambda j[f^{\delta^{-1}(j)}(a_0)]$ is increasing for $0 \leq j < n$. A recursive permutation is in *normal form* if every cycle in its disjoint cycle decomposition is in normal form. Such permutations are more briefly called *normal*.

A permutation where every infinite cycle is normal and every finite cycle $(a_0 \ \dots \ a_{n-1})$ with a_0 as smallest element, has $\lambda j[f^j(a_0)]$ increasing for $0 \leq j < n$, is called *semi-normal*.

Because there is only one way to order the entirety of a cycle into an increasing sequence, there is for any permutation f precisely one (possibly non-recursive) normal permutation f' with $\text{Part } f = \text{Part } f'$. This f' is called the *normal form* of f . The *semi-normal form* of a permutation is defined analogously and also unique.

The less straightforward definition of normality serves the purpose to unify the look of normal cycles regardless of whether they are finite or infinite, which cannot be decided, as is shown in §8. Indeed the idea from Lemma 5.1 can also be made to work with finite cycles if their length is known, as Figure 3 indicates. An archetypal construction of a normal cycle already appeared in Lemma 5.1:

Corollary 5.3. The cycle constructed in the proof of Lemma 5.1 is normal.

Proof. In the nomenclature of the lemma, $f|_A := a\delta \text{succ} \delta^{-1}a^{-1}$. The assertion follows by mere calculation:

$$\begin{aligned} f|_A^{\delta^{-1}(j)} a(0) &= (a\delta \text{succ} \delta^{-1}a^{-1})^{\delta^{-1}(j)} a(0) \\ &= a\delta \text{succ}^{\delta^{-1}(j)} \delta^{-1}a^{-1}a(0) \\ &= a\delta \delta^{-1}(j) = a(j), \end{aligned}$$

which is increasing. Outside of A are only fixed points of f which are normal cycles. \square

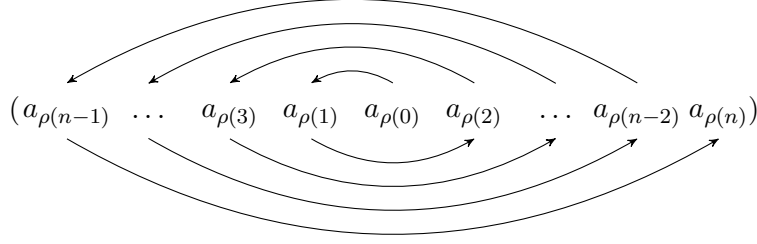


FIGURE 3. Given a cycle $(a_0 a_1 \dots a_n)$ with the rearrangement $a_{\rho(0)} < a_{\rho(1)} < \dots < a_{\rho(n)}$ of the a_i 's into an increasing sequence, use the δ technique from Lemma 5.1 to obtain the displayed normal cycle. The curved arrows indicate the increasing order of the elements, the order from left to right is the cyclic order. It is assumed that n is even for the sake of the example.

Lemma 5.4. There is an algorithm which is uniform in $f \in \mathcal{G}$ and decides for every finite set A , given as a list of its elements, if it is the union of cycles of f , and, if it is not, computes a member of $[A]_f \setminus A$, where $[A]_f := \bigcup_{x \in A} [x]_f$.

Proof. We use the fact that a finite set is a union of cycles of f iff it is closed under application of f . Obtain a list of the members of $X = A \cup f(A)$, which is possible because A was given as such a list. Without loss of generality, these finite lists are without repetition. Then we can decide if $|X| = |A|$. If this is the case, then A is closed under application of f and thus a union of cycles. Otherwise $|X| > |A|$ and A is not a union of cycles. Since evidently $X \subseteq [A]_f$, we can find an element in $[A]_f \setminus A$ by inspecting the list difference $X \setminus A$. \square

The encoding of a finite set as a finite bitstring where the x -th bit is set iff x is a member of the set is called *canonical* [Rog87, § 5.6]. It is easily seen that this canonical encoding is computationally equivalent to the encoding as a finite list. This encoding was essential in the proof to obtain the cardinality of A . It is shown in [Rog87, § 5.6, Theorem XV(b)] that the cardinality of a finite set cannot be computed from a decider for that set, which would have been another candidate for an encoding of finite sets.

Proposition 5.5. Let f be a recursive permutation and let π decide $\Pi = \text{Part } f$. Then there is a recursive f' in normal form with $\text{Part } f' = \text{Part } f$. A program for f' can be computed uniformly effectively from f and π .

Proof. To given x find all $x_0 < x_1 < \dots < x_k = x$ with $x_i \Pi x$. Call the list of these elements A . Using Lemma 5.4, we can determine if A is a union of cycles. If it is, it must be a single cycle because all members of A are in the same cycle. Then the appropriate image of $x = x_k$ can easily be determined, according to Figure 3.

Otherwise Lemma 5.4 gives an element $x^* \in [A]_f \setminus A = [x]_f \setminus A$. Since the x_0, \dots, x_k are all members of the cycle satisfying $x_i \leq x$, it must be $x^* > x$. Search $\hat{x} = \mu \hat{x}[\hat{x} > x \wedge \hat{x} \Pi x]$; this search will terminate as $x^* \Pi x$ and thus $\hat{x} \leq x^*$.

This algorithm provides a way to decide if there are still greater elements than x in its cycle and, in the affirmative case, to find the smallest such element. Figure 3 suggests that this is enough information to carry the construction of the cycle out to its end, in the finite as well as in the infinite case:

$$f'(x) := \begin{cases} x_{k+2}, & k \text{ even and } x_{k+2} \text{ exists,} \\ x_{k+1}, & k \text{ even, } x_{k+1} \text{ exists and } x_{k+2} \text{ does not exist,} \\ x_{k-1}, & k \text{ even and } x_{k+1} \text{ does not exist,} \\ x_0, & k = 1, \\ x_{k-2}, & k \text{ odd and } k > 1. \end{cases}$$

\square

Corollary 5.6. If Π is decidable and permutable, then $\text{Perm } \Pi$ contains exactly one permutation in normal form. \square

Let f be a recursive permutation with decidable cycles. Then $\text{Part } f$ contains, by Corollary 5.6, a recursive permutation f' in normal form, which is the normal form of f . Proposition 2.4 shows that f is effectively conjugate to f' . The converse is also true. Suppose that f is effectively conjugate to its normal form f' . It is to be shown that $\text{Part } f'$ is decidable. The proof exploits the resemblance between a normal cycle and a strictly convex function: the smallest element x_0 of a cycle of f' is characterised by the condition $x_0 \leq \min\{x_0^\pm\}$, where $x^\pm := f^{\pm 1}(x)$. Thus, given any x , an alternating search through the cycle, beginning at x , can be performed to find the smallest element of x 's cycle:

$$\mu x_0[x_0 \equiv_{f'} x] = \xi f'^k(x)[f'^k(x) \leq \min\{f'^{k+1}(x), f'^{k-1}(x)\}],$$

where $\xi f'^k(x)[p(k)]$ is a more suggestive notation for $f'^{\xi k[p(k)]}(x)$.

The alternating search can be replaced by a more intelligent algorithm which further uses the resemblance of normal cycles and strictly convex functions, namely that the direction from any point towards the minimum can be determined by inspecting a neighbourhood of the given point. For any x , compute x^+ and x^- . If, by the test above, x is not the minimum of the cycle, at least one of x^+ and x^- must be smaller than x . Take the smallest of both. If it is x^+ , the minimum can be found in f -positive direction, i.e. the smallest element is of the form $f^k(x)$ with $k > 0$, and if the smaller element is x^- , the minimum is in f -negative direction. By a variant of binary search, the minimum can be found with a number of steps logarithmic in the distance from the starting point to the minimum in the cycle.

Lemma 5.7. The function $\lambda x[\mu x'[x' \equiv_{f'} x]]$ can be computed uniformly effectively in recursive normal permutations f' . \square

If the smallest element of each cycle can be found, the cycles of f' can be decided as shown in Theorem 4.4. To summarise:

Theorem 5.8. Let $f \in \mathcal{G}$. f has decidable cycles iff f is effectively conjugate to its normal form. A decider π for $\text{Part } f$ can be computed from f' . \square

The next results show that instead of effective conjugacy to the normal form, conjugacy to any normal or semi-normal permutation is sufficient.

Lemma 5.9. If a recursive permutation f is effectively conjugate to a normal permutation g via h , i.e. $f = h^{-1}gh$, then it is effectively conjugate to its normal form f' which can be computed from f and h .

Proof. It follows from the conjugation that the normal permutation $g = hfh^{-1}$ is recursive, so that $\text{Part } g \cong \text{Part } f = \text{Part } f'$ is decidable; by Theorem 5.8, a decider γ for $\Gamma = \text{Part } g$ can be found uniformly effectively from g . The proof of Theorem 3.8 shows that the conjugation h is a $(\text{Part } f, \text{Part } g)$ -isomorphism. By Proposition 5.5 we can compute f' from f using the decider $\lambda xx'[h(x)\Gamma h(x')]$ for $\text{Part } f$ which is uniform in γ and h . \square

The semi-normal form has an incompatibility between the structure of finite and infinite cycles. This incompatibility provides additional information: it is possible to decide whether a number lies in a finite or an infinite cycle, uniformly in the permutation. The next lemma shows that this information can be discarded effectively to obtain a normal permutation in the same effective conjugacy class. By Lemma 5.9, it follows that if a permutation f is effectively conjugate to a semi-normal permutation, it is also effectively conjugate to its normal form and therefore decidable.

Lemma 5.10. If f is a recursive semi-normal permutation, then it is effectively conjugate to its normal form f' , which can be computed from f .

Proof. We first note that the structure of a semi-normal permutation allows to decide for every x whether it is in a finite cycle or an infinite one. Let x be given. Again we inspect the neighbours $x^+ = f(x)$ and $x^- = f^{-1}(x)$ of x . x is the least element in the cycle, in case of finite as well as infinite $[x]_f$, iff $x \leq \min\{x^+, x^-\}$. Using alternating search, we can again find the smallest element x_0 in the cycle of x . By computing $x_0, f(x_0), f^2(x_0)$, we can tell if the cycle length is ≤ 2 . If it is, the cycle is, of course, finite. Else the values x_0^-, x_0, x_0^+ are all distinct. If the cycle is infinite, it is in normal form and it must be $x_0 < x_0^- < x_0^+$. If the cycle is finite, it is in semi-normal form and conversely it must be $x_0 < x_0^+ < x_0^-$. This gives a way to decide cycle finiteness.

To construct the normal form f' of f , we determine for an input x if its cycle is finite or infinite. The infinite cycles are already normal because f is semi-normal. If the cycle is finite, we can generate a finite list of all members of the cycle $[x]_f$ by repeated application of f to x . The appropriate image for x to produce a normal cycle can be determined according to Figure 3. \square

Corollary 5.11. If $f \in \mathcal{G}$ is effectively conjugate to a semi-normal permutation, then f has decidable cycles. \square

Semi-normal permutations are studied further in §8. The next subsection deals with permutability criteria for decidable equivalences. Some sufficient conditions even yield permutability by a semi-normal permutation, which provides further motivation for §8.

§6 Permutability. Recall that a decidable equivalence is permutable if its blocks are the orbits of a recursive permutation. Corollary 5.6 already formulated a permutability condition which we restate as

Theorem 6.1. The recursive normal permutations and the decidable and permutable equivalences are in bijection, given by $\phi : g \mapsto \text{Part } g$.

Proof. This mapping is well-defined since the equivalence associated to a recursive normal permutation is decidable by Theorem 5.8. To see bijectivity it suffices to find an inverse mapping. This inverse is the function which associates to every decidable and permutable equivalence its, by Corollary 5.6, unique recursive normal permutation. \square

It is sound to represent a decidable permutable equivalence Π by any pair consisting of a decider π for Π together with an element of $\text{Perm } \Pi$, as these are witnesses for the asserted properties of the equivalence. Under this representation, the bijection and its inverse are computable. In one direction, any recursive normal f' maps to the pair (π, f') which represents $\text{Part } f'$. As Theorem 5.8 shows, π can be obtained from f' . In the other direction, (π, f) maps to the normal form f' of f , which is computable from π and f by Proposition 5.5.

A closer inspection of the proof of Proposition 5.5 shows that the algorithm to construct the normal element can be reformulated to rely on the decidability of Π and a way to determine for any x if there is a greater element in its block. A permutation $f \in \text{Perm } \Pi$ provided such a method in the proof of Proposition 5.5. We proceed to show the converse: if such a method is available, there must be an $f \in \text{Perm } \Pi$:

Theorem 6.2. Let Π be decidable via π . Π is permutable iff the predicate $\rho(x) = [\exists x' > x : x' \Pi x]$ is recursive. Given π , such a function ρ can be constructed from the normal permutation for Π , and therefore, in the presence of π , from any member of $\text{Perm } \Pi$, and vice versa.

Proof. “ \Rightarrow ”: If $\text{Perm } \Pi \neq \emptyset$ there is an $f' \in \text{Perm } \Pi$ in normal form. Let x be given. By Lemma 5.7 we can find the smallest number x_0 in $P(x)$. The function $a(j) := f'^{\delta^{-1}(j)}(x_0)$ enumerates $P(x)$ in increasing order for $0 \leq j < |P(x)|$. We can obtain the smallest index $k = k(x)$ such that $a(k) = x$.

Suppose $P(x)$ is finite of length $n \in \mathbb{N}^+$. Since a is increasing for its first n arguments, we have $a(n) \leq a(n-1)$, because $a(n) \in P(x)$ and $a(n-1)$ is the greatest element therein. By definition $0 \leq k(x) < n$. Now, x is the greatest element in the finite block iff $k(x) = n-1$

iff $a(k(x) + 1) \leq ak(x)$, the latter of which can be checked without knowing n . If the block is infinite, there cannot be a greatest element and indeed $a(k(x) + 1) > ak(x)$ will always hold. Therefore $[\exists x' > x : x' \Pi x] = [ak(x) < a(k(x) + 1)]$ gives a uniform way to compute ρ .

“ \Leftarrow ”: The construction is the same as in the proof of Proposition 5.5, except that ρ is used instead of the function f there to decide $[\exists x' > x : x' \Pi x]$. \square

The next goal is to obtain a non-permutable equivalence relation. This can be achieved by encoding the computation of Turing machines well enough so that decidability of the criterion in Theorem 6.2 implies decidability of the Halting Problem. The equivalence is defined over codings of pairs $\langle x, n \rangle$ where x is a program and n a step counter in the computation $\varphi_x(x)$. This setting reveals a flaw of Theorem 6.2: deciding $\langle x, n \rangle > \langle x', n' \rangle$ requires knowledge of the coding $\lambda x n [\langle x, n \rangle]$. It would suffice for the current purpose to fix the *standard coding* of pairs $\langle x, y \rangle := \frac{1}{2}(x^2 + 2xy + y^2 + 3x + y)$ [Rog87, p. 64], because it is strictly increasing in its second parameter. Such a fixation is unpleasant and defining permutations over tuples is a useful tool in general. We provide at least a variant of Theorem 6.2 which is independent of the coding of pairs and deals with the kind of equivalence that is later considered in Proposition 6.5 and Lemma 8.2.

Definition 6.3. An infinite family Π_z , $z \in \mathbb{N}$, of decidable equivalences is *uniformly decidable* if there is a recursive function ψ such that $\psi(z, x, x') = [x \Pi_z x']$. In this case the *coproduct equivalence* Π defined by

$$\langle z, x \rangle \Pi \langle z', x' \rangle :\Leftrightarrow z = z' \wedge x \Pi_z x'$$

is again decidable.

The blocks of a coproduct Π of a family Π_z are of the form $P(\langle z, x \rangle, \Pi) = \{\langle z, x' \rangle : x' \in P(x, \Pi_z)\}$, i.e. the blocks of Π_z are prefixed by z , to make all blocks across the family disjoint, and then Π is the partition consisting of all these blocks.

Corollary 6.4. Let Π be the coproduct of a uniform family Π_z of decidable equivalences. Π is permutable iff the predicate $\rho\langle z, x \rangle = [\exists x' > x : x' \Pi_z x]$ is recursive.

Proof. “ \Rightarrow ”: Let $f \in \text{Perm } \Pi$. Uniform in z , we obtain functions $f_z = \lambda x [\pi_2 f\langle z, x \rangle] \in \text{Perm } \Pi_z$, where $\pi_2 \langle z, x \rangle := x$ denotes the projection of a pair on its second component. By Theorem 6.2, there is a function ρ_z for Π_z , which can be computed from f_z and a decider π_z for Π_z , by Proposition 5.5 and Theorem 6.2. f_z and π_z , in turn, can be computed uniformly effectively from f , z and a uniform decider ψ for the family Π_z . Thus $\lambda \langle z, x \rangle [\rho_z(x)]$ is recursive and has the desired property.

“ \Leftarrow ”: For any fixed z , $\lambda x [\rho\langle z, x \rangle]$ is a ρ function for Π_z as in Theorem 6.2. This gives a permutation $f_z \in \text{Perm } \Pi_z$ which can be constructed uniformly effectively from ρ , z and ψ . Then the function $f\langle z, x \rangle := \langle z, f_z(x) \rangle$ is a recursive permutation and moreover a member of $\text{Perm } \Pi$. \square

Proposition 6.5. There exists a decidable equivalence which is not permutable.

Proof. Define

$$r'_x(n) := [\varphi_x(x) \text{ halts after } \leq n \text{ steps}],$$

$$r_x(n) := \begin{cases} \mathbf{t}, & n = 0, \\ r'_x(n-1), & n > 0. \end{cases}$$

Each r_x is a recursive function which induces a decidable equivalence Π_x , as per Proposition 1.4. Indeed this family of equivalences is uniformly decidable because $[r_x(n) = r_x(n')]$ can be decided uniformly in x, n, n' , using a universal Turing machine. Let Π be the coproduct of this family.

Assume that Π is permutable so that Corollary 6.4 yields a recursive function $\rho\langle x, n \rangle = [\exists n' > n : n' \Pi_x n]$. In particular for $n = 0$, we can decide $[\exists n' \geq 1 : r_x(n') = \mathbf{t}]$ uniformly in x , i.e. whether $\varphi_x(x)$ halts eventually. This *contradicts* the undecidability of the Halting Problem. \square

Theorem 6.6. Let Π be decidable via π and let there be a recursive function ρ such that

$$\rho(x) = \begin{cases} 0, & |P(x)| = \infty, \\ |P(x)|, & \text{else.} \end{cases}$$

Then Π is permutable by a semi-normal permutation which can be constructed from π and ρ .

Without the last addition that there be a semi-normal permutation in $\text{Perm } \Pi$, a proof would have been immediate from Theorem 6.2. Theorem 8.5 in §8 shows that the converse of Theorem 6.6 holds, too. Together with other results from §8, this shows that there are permutable equivalences which lack a semi-normal element, which is why Theorem 6.6 cannot be inferred from Theorem 6.2 and needs a separate proof.

Proof. Let x be given. With a decider for Π , a decider for $P(x)$ can be found uniformly in x , by Lemma 1.2. Using ρ , it is decidable whether $P(x)$ is finite or infinite. If it is finite, all elements of $P(x)$ can be found by testing $x' \in P(x)$ until $\rho(x)$ members are found. These numbers can easily be arranged into a finite semi-normal cycle. If the cycle is infinite, it is an infinite decidable set and Lemma 5.1 plus Corollary 5.3 provide a method to construct the infinite semi-normal (i.e. normal) cycle. \square

Corollary 6.7. If Π is decidable, then each of the following conditions is sufficient for the permutability of Π by a semi-normal element:

- (a) Π has only finitely many blocks,
- (b) Π has only blocks of the same cardinality (including \aleph_0).

Proof. We show that there is a recursive function ρ as in Theorem 6.6.

(a) Let $\Pi = \{P_1, \dots, P_n\}$, $p_i = |P_i|$ and $y_i \in P_i$ arbitrary representatives. Using $y(x) := \mu y[y \in P(x)]$ we can find the unique index i such that $y(y_i) = y(x)$, then set $\rho(x) := p_i$. The program for ρ only needs to include the subroutine $y(x)$ and the finitely many constants y_i and p_i , $1 \leq i \leq n$.

(b) The function ρ returning the cardinality of a block is constant and therefore recursive. \square

It is easy to construct counterexamples to the converses of Corollary 6.7 and Proposition 4.2 from §4. There is a recursive semi-normal permutation, implying decidable cycles, with infinitely many infinite cycles and one finite cycle. It is therefore a simultaneous counterexample to the converses of the corollary and the proposition. Consider the partition Π whose blocks consist of all numbers ≥ 2 which have the same smallest prime factor, and 0 and 1 form another block together. This is a decidable partition. By Theorem 6.6 (with $\rho(0) := \rho(1) := 2$ and $\rho(x) := 0$ else) we see that Π has a semi-normal $f \in \text{Perm } \Pi$ which is a simultaneous counterexample.

It was shown in the proof of Lemma 5.10 that the cycle structure of a recursive semi-normal permutation makes it possible to decide the finiteness of the cycle of any given number x . Theorem 6.6 has a similar direction and also involves semi-normal permutations. §8 deals with cycle finiteness and its relation to semi-normality more systematically.

§7 An order-theoretic characterisation of $\text{Perm } \Pi$. This section characterises the elements of $\text{Perm } \Pi$ for a fixed decidable equivalence Π as the maximal elements with respect to cycle inclusion inside a normal subgroup of $\text{Aut } \Pi$. The merit of this theorem is that Recursion Theory only appears in the setting; the characterisation itself makes no use of the language of computability. For the basic order-theoretic notions needed here, see e.g. [Sch16].

Let $\mathcal{I}\Pi := \{f \in \text{Aut } \Pi : f(P) = P \ \forall P \in \Pi\}$ denote the set of *block-wise identical* permutations in $\text{Aut } \Pi$. Any $f \in \text{Perm } \Pi$ achieves $[x]_f = P(x, \Pi)$, which implies $f(P) = P$ for every block $P \in \Pi$. This means $\text{Perm } \Pi \subseteq \mathcal{I}\Pi$ and furthermore $\mathcal{I}\Pi$ is a normal subgroup of $\text{Aut } \Pi$. For an alternative view on $\mathcal{I}\Pi$, define a *refinement* of an equivalence Π to be an equivalence Π' such that $x\Pi'x' \Rightarrow x\Pi x'$. In this case we write $\Pi' \leq \Pi$. No decidability requirements are attached to this notion and, in this subsection, the symbol $\text{Perm } \Pi$ shall not imply that Π is decidable. Then $\mathcal{I}\Pi = \bigcup_{\Pi' \leq \Pi} \text{Perm } \Pi'$. The finest and coarsest equivalences, e.g., yield $\mathcal{I}\{\{0\}, \{1\}, \dots\} = \{\text{id}\}$ and $\mathcal{I}\{\mathbb{N}\} = \mathcal{G}$, but the automorphism group is \mathcal{G} in both cases.

The relation

$$f \lesssim g :\Leftrightarrow \forall x : [x]_f \subseteq [x]_g$$

is a preorder on \mathcal{II} . It becomes antisymmetric if the permutations are collapsed to their cycle equivalences, i.e. if one disregards the sequence of elements in the cycles of a permutation. By $f \gtrsim g$ we mean $f \lesssim g$ and $g \not\lesssim f$. A permutation $f \in \mathcal{II}$ is *maximal* if there is no $g \in \mathcal{II}$ such that $f \gtrsim g$, i.e. f is an upper bound on all elements it is comparable to. The set of maximal elements of \mathcal{II} is written $\max \mathcal{II}$.

Theorem 7.1. If Π is decidable, then $\text{Perm } \Pi = \max \mathcal{II}$.

Proof. First suppose Π is permutable. Let $f \in \mathcal{II}$ and $g \in \text{Perm } \Pi$ be arbitrary. From $[x]_f \subseteq P(x)$ it follows that the restriction $f|_P$ is a permutation of P for every block $P \in \Pi$. This means that P is a union of cycles of f . Since $g \in \text{Perm } \Pi$, we know $[x]_g = P(x)$ and thus $f \lesssim g$. This shows that every element of $\text{Perm } \Pi$ is an upper bound on \mathcal{II} and in particular maximal. On the other hand, if g is maximal in \mathcal{II} and Π is permutable, there exists an $f \in \text{Perm } \Pi$ and by the first part of the proof $g \lesssim f$. Maximality of g then implies $f \lesssim g$, i.e. $P(x) = [x]_f \subseteq [x]_g \subseteq P(x)$ and it follows equality everywhere and $g \in \text{Perm } \Pi$.

If Π is non-permutable, we wish to show that there are no maximal elements. Given any $f \in \mathcal{II}$, we construct a permutation in \mathcal{II} which is comparable to and strictly greater than f . Since $f \notin \text{Perm } \Pi$, there is an z such that $[z]_f \subsetneq P(z)$. Since $P(z)$ is a union of cycles of f , $P(z)$ must be the union of at least two cycles of f . Because Π is decidable, $P(z)$ is a recursive set. The equivalence $\{P(z), \overline{P(z)}\}$ is evidently decidable and has finitely many blocks, which is a sufficient permutability condition. By Corollary 6.7 we obtain a permutation c , one of whose cycles is $P(z)$. Then

$$f'(x) := \begin{cases} f(x), & x \notin P(z), \\ c(x), & x \in P(z), \end{cases}$$

replaces the multitude of cycles in f , which make up $P(z)$, by a single cycle. f' is in \mathcal{II} and strictly greater than f . \square

The proof shows the slightly stronger statement that every member of $\text{Perm } \Pi$ is not only a maximum of \mathcal{II} but also an *upper bound*, i.e. it is maximal and comparable to every member of \mathcal{II} . We also remark that finding the $f' \gtrsim f$ in the second part of the proof was an instance of permutability. We had to find a permutable, not necessarily decidable equivalence $\text{Part } f'$ such that $\text{Part } f \leq \text{Part } f' \leq \Pi$. The proof above shows that it is not hard to order individual blocks of an equivalence into a single cycle of a recursive permutation, at least if the block is decidable and its size is known (cf. Corollary 6.7). The hard part of permutability is ordering infinitely many blocks into cycles simultaneously. The proof gives the following picture of the order in \mathcal{II} if Π is decidable but not permutable: every chain in \mathcal{II} with a maximal element can be extended by adding a strictly greater element which assembles one further block of Π into a single cycle. From this perspective at least, chains grow along the blocks of Π , and Π has infinitely many blocks as it is not permutable.

Corollary 7.2. $\text{Perm} = \bigcup_{\Pi \text{ dec.}} \max \mathcal{II}$. \square

IV. CYCLE FINITENESS AND UNSOLVABLE PROBLEMS

This last section proves negative answers to algorithmic questions surrounding Perm . The *cycle finiteness problem* is introduced and it is shown that it is in general unsolvable for permutations with decidable cycles. The subset Perm_{CF} of Perm where cycle finiteness is decidable is characterised by semi-normal permutations. It is shown that cycle decidability and cycle finiteness problems in \mathcal{G} are intertwined by one-one reductions and that the maximum one-one degree of either problem is the Halting Problem.

Furthermore it is shown that conjugacy in Perm is undecidable, that Perm is not enumerable, and that it is not closed under multiplication. Lastly while Perm is closed under multiplication with finitary permutations, it can be shown that there is no constructive proof of this fact,

assuming that a finitary permutation a is encoded as a list of transpositions, whose product is a , and each transposition is encoded as an ordered pair. A constructive proof can be given for Perm_{CF} .

§8 Cycle finiteness. We begin by constructing a permutation for which it is undecidable if numbers lie in a finite or an infinite cycle. Such a permutation has already been described in [Leh09], by encoding the Halting Problem for Turing machines into the cycle length. Indeed cycle finiteness is the prototype of Collatz' original problem, which is the motivation of Lehtonen's paper. His permutation has the additional property that it can be described using a case distinction on a decidable partition with 5 blocks, and each case has the form of an affine-linear function. The construction below will use the general theory developed so far, which makes it swift but does not yield similar properties.

Definition 8.1. For $f \in \mathcal{G}$, the *cycle finiteness problem* of f is the decision problem

$$\text{CF}(f) := \{x : |[x]_f| < \infty\}.$$

The subset of Perm consisting of permutations with decidable cycle finiteness problem is denoted Perm_{CF} .

We use a slight modification of the proof of Proposition 6.5. Define the x -indexed family of recursive predicates $r'_x(n)$ by

$$r'_x(n) = [\varphi_x(x) \text{ halts after } \leq n \text{ steps}].$$

The family Π_x of equivalences which correspond to these recursive functions via Proposition 1.4 is uniformly decidable by a universal Turing machine, which makes their coproduct Π a decidable equivalence. The interpretation of the blocks of Π is as follows: $\langle x, n \rangle$ and $\langle x', n' \rangle$ are in the same block iff they belong to the same program, $x = x'$, and either both computations (after n and n' steps) did not halt yet or both halted.

We want to show that $\rho\langle x, n \rangle = [\exists n' > n : r'_x(n) = r'_x(n')]$ is recursive in order to apply Corollary 6.4. To given $\langle x, n \rangle$, simulate the computation $\varphi_x(x)$ for $n + 1$ steps. If it halts after $\leq n$ steps, it also halts after $\leq n + 1$ steps, so $r'_x(n) = r'_x(n + 1)$ and $\rho\langle x, n \rangle = \mathbf{t}$. If it halts at the $(n + 1)$ -st step, then $r'_x(n') \neq r'_x(n)$ for all $n' > n$ and $\rho\langle x, n \rangle = \mathbf{f}$. The remaining case is that the computation did not halt after $n + 1$ steps, in which case it did not halt after $\leq n$ steps either, and $r'_x(n) = r'_x(n + 1)$, $\rho\langle x, n \rangle = \mathbf{t}$. By Corollary 6.4, there is a member $g \in \text{Perm } \Pi$.

Assume, we could decide $|[\langle x, n \rangle]_g| < \infty$, for every pair $\langle x, n \rangle$. Let a program x be given. Then we could decide whether $\langle x, 0 \rangle$ lies in a cycle of finite length, which is the same as deciding whether the computation of $\varphi_x(x)$ halts eventually. This *contradicts* the undecidability of the Halting Problem.

Lemma 8.2. There is a $g \in \text{Perm} \setminus \text{Perm}_{\text{CF}}$. □

Define the decision problem CF^* as follows: given $f \in \text{Perm}$, a decider π for the cycles of f and a number x , it is to decide whether $|[x]_f| < \infty$. A fortiori, this problem is recursively unsolvable. The *diagonal* problem ΔCF^* of CF^* asks, given a program which computes a permutation with decidable cycles, and a decider for the cycles, if that program itself is in a finite or an infinite cycle of the permutation. This problem may be thought of as the Perm version of the one-parameter Halting Problem $K := \{x : \varphi_x(x) \text{ halts}\}$. Using the Recursion Theorem, one can reduce the seemingly more general problem CF^* to its diagonal and obtains

Corollary 8.3. ΔCF^* is recursively unsolvable. □

We can improve upon the inclusion $\mathcal{G}_1 \subseteq \text{Perm}$ from Proposition 4.2, by using essentially the same technique as in that proof.

Proposition 8.4. Every permutation with finitely many infinite cycles has decidable cycle finiteness problem, i.e. $\mathcal{G}_1 \subseteq \text{Perm}_{\text{CF}}$.

Proof. Let $f \in \mathcal{G}_1$ and x_1, \dots, x_n be a system of representatives for the infinite cycles of f . By Proposition 4.2, f has decidable cycles. Given a number x , we can decide if x belongs to any of the cycles $[x_1]_f, \dots, [x_n]_f$. This is the case iff $|[x]_f| = \infty$. \square

Theorem 6.6 stated that if a decidable equivalence Π possesses a recursive function ρ which returns to each x the size of $P(x)$, or 0 if $P(x)$ is infinite, then Π is permutable by a semi-normal element. We see now that not every decidable permutable equivalence has such a function ρ . Take the function g from Lemma 8.2: Part g is decidable and permutable. If such a function ρ existed for Part g , the computable function $\lambda x[\rho(x) \neq 0]$ would decide $\text{CF}(g)$, which is impossible. The next theorem links cycle finiteness to the conjugacy classes of semi-normal permutations:

Theorem 8.5. Let $f \in \text{Perm}$. Then the following statements are equivalent:

- (a) $\text{CF}(f)$ is decidable,
- (b) there is a ρ function as in Theorem 6.6 for Part f , and
- (c) f is effectively conjugate to its semi-normal form.

Proof. “(a) \Rightarrow (b)”: Write $\Pi = \text{Part } f$ and $P(x) = P(x, \Pi)$ as usual. By the assumption we can compute $[|P(x)| < \infty]$. Let x be given. If $|P(x)| = \infty$, then report $\rho(x) := 0$. Else $P(x) = [x]_f$ is a finite set which we can enumerate by powers of f on x : determine $n = \mu n[n \geq 1 \wedge f^n(x) = x]$. Then n is the length of $[x]_f$ and we report correctly $\rho(x) := n$.

“(b) \Rightarrow (c)”: Given ρ , Theorem 6.6 yields a recursive semi-normal element f' in $\text{Perm } \Pi$, which is the semi-normal form of f . Both permutations are recursive and hence effectively conjugate by Proposition 2.4.

“(c) \Rightarrow (a)”: Let f' be the semi-normal form of f . Because f and f' are effectively conjugate, f' is recursive. The proof of Lemma 5.10 shows that a recursive semi-normal permutation can be used to solve its own cycle finiteness problem. Since $\text{Part } f = \text{Part } f'$, it follows that $\text{CF}(f) = \text{CF}(f')$ is decidable. \square

Corollary 8.6. Perm_{CF} is the union of effective conjugacy classes of recursive semi-normal permutations. \square

Corollary 8.7. The recursive semi-normal permutations and the decidable, permutable equivalences with decidable block finiteness problem are in bijection via $g \mapsto \text{Part } g$. \square

§9 Conjugacy and enumerability. As shown by the Theorems 3.8, 5.8 and 8.5, conjugacy in Perm is equivalent to the solvability of certain decision problems. We proceed to prove that conjugacy, and therefore the solvability of these problems, cannot be decided.

To a $g \in \text{Perm}$ define the problem $\text{CONJ}(g)$, which asks, given $f \in \text{Perm}$ and a decider for its cycles, to decide whether $f \sim g$. We describe a permutation g for which this problem is unsolvable. This immediately implies that conjugacy between two given members of Perm can in general not be decided.

Theorem 9.1. There is a $g \in \text{Perm}$ such that $\text{CONJ}(g)$ is recursively unsolvable.

Proof. Define g to be

$$g(x) := \begin{cases} x, & x \equiv 1 \pmod{2}, \\ 0, & x = 2, \\ x - 4, & x \equiv 2 \pmod{4}, x \neq 2, \\ x + 4, & x \equiv 0 \pmod{4}, \end{cases}$$

i.e. $g = (\dots 6 \ 2 \ 0 \ 4 \ 8 \ \dots)$.

The proof is by (truth-table) reduction of CF^* . Let f, π and x be given. We define another permutation f' in a way that if $[x]_f$ is finite, f' consists only of finite cycles, whereas if $[x]_f$ is infinite, all cycles in f' are either 1-cycles or infinite and there are infinitely many 1-cycles and one infinite cycle. Therefore f' and g have the same cycle type iff $[x]_f$ is infinite. Since f' and

g have finitely many infinite cycles, cycle type equality is equivalent to effective conjugacy of f' and g , by Corollary 4.3 and Theorem 3.8. This shows $|[x]_f| < \infty \Leftrightarrow f' \not\sim g$. It therefore suffices to construct a permutation f' and a decider π' for its cycles, uniformly in f, π, x , such that f' has the same cycle type as g iff $[x]_f$ is infinite.

Let $k(x') := \xi k[f^k(x) = x']$. Define the following equivalence relation Π' : every $x' \notin [x]_f$ is alone in a block and so is every $x' \in [x]_f$ with $k(x') \equiv 1 \pmod{2}$. The remaining numbers $x' \in [x]_f$ with $k(x') \equiv 0 \pmod{2}$ form a block together. This equivalence is evidently decidable and a decider π' can be computed uniformly in f, π and x . Because f and π are available, we can compute a ρ function for Part f , as in Theorem 6.2. This ρ function can be used to define a ρ function ρ' for Π' in the following manner: let y be given. If $y \notin [x]_f$ or $y \in [x]_f$ and $k(y) \equiv 1 \pmod{2}$, then $|P(y, \Pi')| = 1$ and there is no greater element in the same block. Now assume $y \in [x]_f$ and $k(y)$ even. We have to decide whether there is a $y' > y$ with $y' \in [x]_f$ and $k(y')$ even. First, using ρ , we can determine if there is a $y' > y$ which is also in $[x]_f$. If not, $\rho'(y) := \mathbf{f}$. Otherwise we can find the smallest value $y' > y$ with $y' \in [x]_f$. If $k(y')$ is even, we are done and report $\rho'(y) := \mathbf{t}$. Otherwise we repeat the procedure with y' .

This algorithm lists all elements of $[x]_f$ which are greater than y in order until one with even k index is found or the cycle is exhausted. Thus if the algorithm terminates, we have either witnessed the existence of a greater element than y in $P(y, \Pi')$ or we have verified that all elements in $[x]_f \supseteq P(y, \Pi')$ which are greater than y are not in $P(y, \Pi')$. So if the algorithm terminates, it yields a correct answer. It remains to prove that it always terminates. If $[x]_f$ is finite, the algorithm halts at the latest after the cycle is exhausted. If $[x]_f$ is infinite, then there are infinitely many numbers with even k index, and thus arbitrarily large ones; the algorithm will eventually find one and terminate. With ρ' , Theorem 6.2 gives, still uniformly in f, π and x , a permutation $f' \in \text{Perm } \Pi'$. If $[x]_f$ is finite, all cycles in f' are finite. If $[x]_f$ is infinite, f' consists of infinitely many 1-cycles and one infinite cycle. This completes the proof. \square

The second task treated in this subsection is enumerability. It may be useful for various constructions to compute an exhaustive list of all the members of Perm . This is shown impossible here, i.e. there is no partial recursive function ρ such that:

- (1) $\forall x \in \text{dom } \rho : \varphi_{\rho(x)} \in \text{Perm}$, and
- (2) $\forall f \in \text{Perm} \exists x \in \text{dom } \rho : f = \varphi_{\rho(x)}$.

Theorem 9.2. Perm is not recursively enumerable.

There are multiple accessible proofs of this theorem. The first is an obvious but somewhat technical diagonalisation. The second proof is based on Corollary 3.9 and Kent's result [Ken62, Thm. 2.1] about the composition series of \mathcal{G} . From these two follows that the subgroup of \mathcal{G} generated by Perm is already all of \mathcal{G} . If Perm was enumerable, then so would be its group closure, but this *contradicts* the inenumerability of \mathcal{G} , [Rog87, Ex. 4-6]. We give another short proof based on a result by van Leeuwen:

Proof. By [vL15, Thm. 4], no recursively enumerable set of partial recursive functions with infinite domains can contain all involutions. All members of Perm are permutations and have infinite domains, but by Lemma 4.1, Perm contains all recursive permutations with only finite cycles, in particular all involutions. It follows that Perm is not enumerable. \square

§10 Difficulty of cycle decidability. [Ken62] provides a tool to obtain permutations with particularly difficult cycle structure. Let $W_x := \text{dom } \varphi_x$ denote the standard numbering of recursively enumerable sets. Recall from [Rog87, § 7.3] that a set P is *productive* if there is a partial recursive ψ , such that whenever $W_x \subseteq P$ it follows that $\psi(x)$ is convergent and $\psi(x) \in P \setminus W_x$. The function ψ is called a *productive function* for P . A set C is *creative* if it is a recursively enumerable complement of a productive set. One example of a creative set is the Halting Problem $K := \{x : x \in W_x\}$ whose complement has the identity as a productive function.

Theorem ([Ken62, Thm. 1.3]). Let C be a creative set. There is a recursive permutation k composed of infinitely many infinite cycles, one of which is C .

We remark that the proof of Theorem 1.3 in [Ken62] is largely based on his Lemma 1.4, which does not hold as stated there. It states that if A is a non-empty recursively enumerable set, then there is a recursive permutation with infinitely many infinite cycles, one of which is the cylinder $A \times \mathbb{N} \subseteq \mathbb{N}$ [Rog87, § 7.6]. However, for $A = \mathbb{N}$, the cylinder $\mathbb{N} \times \mathbb{N} = \mathbb{N}$ and if one of the cycles of the permutation is \mathbb{N} , it cannot have infinitely many cycles. The proof given in [Ken62] shows the assertion under the additional assumption that A has infinite complement. This result is sufficient to infer his Theorem 1.3, as creative sets necessarily have infinite complements.

With this theorem we obtain a recursive permutation k with a creative cycle. The complement of this cycle is productive. By definition, a productive set is not recursively enumerable. On the other hand, if $f \in \text{Perm}$, then every cycle of f is a recursive set, so the complement of every cycle is recursive, too. It follows

Lemma 10.1. There is a permutation $k \in \mathcal{G} \setminus \text{Perm}$ with infinitely many infinite cycles, one of which is the Halting Problem K . \square

Alternatively, [Hig90, Thm. 3.1] shows that there is a recursive permutation with infinitely many infinite cycles and no finite cycles, all of whose transversals are immune [Rog87, § 8.2]. Since an immune set is not recursively enumerable, it follows by Theorem 4.4 that this permutation has undecidable cycles.

We note that Kent's and Higman's constructions produce a permutation with infinitely many infinite cycles. Indeed, a permutation with finitely many infinite cycles is necessarily in Perm by Proposition 4.2.

The fact that there are permutations with undecidable cycles raises the question of how difficult cycle decidability problems for recursive permutations can become. The following proposition determines the maximal one-one degree of cycle decidability problems in \mathcal{G} . For the definition of reducibilities and degrees, the reader is referred to [Rog87, §§ 6ff.].

Proposition 10.2. For every $f \in \mathcal{G}$, the cycle decidability problem of f is one-one reducible to the Halting Problem, and there exists an $f' \in \mathcal{G}$ such that the Halting Problem is one-one reducible to the cycle decidability of f' .

Proof. Let $f \in \mathcal{G}$ be arbitrary. With an oracle for the Halting Problem K we can check uniformly in x, x' whether the function $\lambda w[\xi k[f^k(x) = x']]$ halts on its own encoding (or any other number because the function ignores its argument). This is the case iff $x \equiv_f x'$. The mapping of $\langle x, x' \rangle$ to a program for $\lambda w[\xi k[f^k(x) = x']]$ can be chosen to be strictly increasing in the numerical value of $\langle x, x' \rangle$ which makes it a one-one reduction.

For the opposite direction, Lemma 10.1 shows that there is a recursive permutation $k \in \mathcal{G}$ of which one cycle is the Halting Problem. Fix a program x_0 such that $\varphi_{x_0}(x_0)$ halts. Then the Halting Problem can be solved by $x \in K \Leftrightarrow x \equiv_k x_0$ which is a one-one reduction. \square

In previous sections we have used conjugacy and normal forms to characterise the solvability of cycle decidability and cycle finiteness problems. An interesting fact is that these two classes of problems in \mathcal{G} are inter-reducible. To prove this, a technical lemma is needed:

Lemma 10.3. For every $g \in \mathcal{G}$ there is a $g' \in \mathcal{G}$ and an embedding $j : \mathbb{N} \rightarrow \mathbb{N}$ such that $||[x]_g| < \infty \Leftrightarrow |[j(x)]_{g'}| < \infty$ and all cycles of g' which intersect $\text{rng } j$ are infinite or of odd length.

Proof. The idea is to create for each x a copy of its cycle, double its length and then add one further element to it. This preserves cycle finiteness and makes every finite cycle odd. Define $j(x) := \langle x, x, 0 \rangle$ and g' by

$$\begin{aligned} \langle x, x, 0 \rangle &\mapsto \langle x, x, 1 \rangle \mapsto \langle x, x, 2 \rangle \mapsto \langle x, g(x), 0 \rangle \\ \langle x, y, 0 \rangle &\mapsto \langle x, y, 1 \rangle \mapsto \langle x, g(y), 0 \rangle, \quad y \neq x \end{aligned}$$

and fix every triple which does not match the decidable patterns above. Clearly g' is a permutation and every cycle which contains some $\langle x, x, 0 \rangle$ is either infinite or of odd length. Since the cycle structure of g is transferred into the second component and merely stretched by the third component in the definition of g' , we have $|[x]_g| < \infty \Leftrightarrow |[j(x)]_{g'}| < \infty$ for all x . $|[j(x)]_{g'}|$ is either infinite or odd. \square

Theorem 10.4. In \mathcal{G} , the classes of cycle decidability and cycle finiteness problems are one-one inter-reducible, in the following sense:

- (i) For every $f \in \mathcal{G}$ there is a $g \in \mathcal{G}$ and an embedding j such that $x \equiv_f y \Leftrightarrow |[j\langle x, y \rangle]_g| < \infty$.
- (ii) For every $g \in \mathcal{G}$ there is an $f \in \mathcal{G}$ and two embeddings j, j' such that $|[x]_g| < \infty \Leftrightarrow j(x) \equiv_f j'(x)$.

Proof. (i) For every pair x, y define the relation

$$i \Pi_{x,y} j \Leftrightarrow i = j \vee \forall k \in \mathbb{Z}, |k| \leq \max\{i, j\} : f^k(x) \neq y.$$

$\Pi_{x,y}$ is easily seen to be reflexive, symmetric and transitive, and thus an equivalence. Indeed $\Pi_{x,y}$ is a family of uniformly decidable equivalence relations indexed by $\langle x, y \rangle$. Let Π denote their coproduct. The relation $\Pi_{x,y}$ is defined in such a way that for any i there exists an $i' > i$ with $i' \Pi_{x,y} i$ iff $(i+1) \Pi_{x,y} i$. By the uniform decidability of the relations we immediately obtain a uniform ρ function as in Corollary 6.4 for this family. Application of this corollary gives $g \in \text{Perm } \Pi$. Observe that the block of 0 in $\Pi_{x,y}$ contains infinitely many elements iff $f^k(x) \neq y \forall k \in \mathbb{Z}$. The embedding is $j\langle x, y \rangle = \langle x, y, 0 \rangle$.

(ii) By Lemma 10.3 we find a g' and an embedding j such that $[j(x)]_{g'}$ is finite iff $[x]_g$ is, and $[j(x)]_{g'}$ is either infinite or of odd length. If $[j(x)]_{g'}$ is infinite, $j(x)$ and $g'j(x)$ cannot be in the same cycle of g'^2 . On the other hand, if $[j(x)]_{g'}$ is finite, its length is odd. Application of g' on such a cycle $[j(x)]_{g'}$ imposes the structure of a finite cyclic group of odd order on the cycle with the group operation $g'^k j(x) \cdot g'^l j(x) := g'^{k+l} j(x)$. Since 2 is coprime to the order of this group, $g'^2 j(x)$ is a generator and $[j(x)]_{g'^2}$ must contain $g'j(x)$. We have shown that $|[x]_g| < \infty$ iff $|[j(x)]_{g'}| < \infty$ iff $j(x) \equiv_{g'^2} g'j(x)$, thus set $f = g'^2$ and $j' = g'j$. \square

§11 Products in Perm. Theorem 3.8 shows that effective conjugacy and effective cycle type equality are equivalent in Perm. Recall that effective cycle type equality of two permutations f, g means computable isomorphy of their respective equivalences $\text{Part } f, \text{Part } g$. To formulate this theorem constructively, there must be constructive representations of these equivalences, i.e. they must be decidable and represented by their deciders, or by an equivalent means, as presented in §1. In this way Perm is the maximal domain for this theorem. As discussed in the introduction, the theorem is a constructive analogue of the well-known theorem that conjugacy in a symmetric group is equivalent to cycle type equality, where Perm plays the role of the full symmetric group. This raises the question about the algebraic structure of Perm. $\mathcal{G} \supseteq \text{Perm}$ imposes its group operation on Perm and it is evident that (a) $\text{id} \in \text{Perm}$ and (b) if $f \in \text{Perm}$ then $f^{-1} \in \text{Perm}$.

This subsection proves that multiplicative closure fails. The proof uses structural results of [Ken62] to infer the existence of $f, g \in \text{Perm}$ such that $fg \notin \text{Perm}$ without constructing them. The present proof could have been given after Corollary 3.9 was established. A second theorem contains a positive result in the direction of multiplicative closure, namely that Perm is closed under multiplication from left and right with *finitary* permutations. However, it is also shown that there is a fixed permutation $g \in \text{Perm}$ such that cycle decidability of finitary products cannot be witnessed uniformly. The cycle finiteness problem introduced in §8 plays an important role in the characterisation of the cases where computing the decider is possible.

By Corollary 3.9, Perm is a normal subset in \mathcal{G} , so that the subgroup $\langle \text{Perm} \rangle$ generated by Perm is a normal subgroup of \mathcal{G} . Clearly Perm contains all finitary permutations and a non-finitary one, e.g. $\delta \text{succ } \delta^{-1} = (\dots \ 5 \ 3 \ 1 \ 0 \ 2 \ 4 \ 6 \ \dots)$ whose decider is trivial.

Theorem ([Ken62, Thm. 2.1]). If N is a normal subgroup of \mathcal{G} that contains a permutation with infinite support, then $N = \mathcal{G}$.

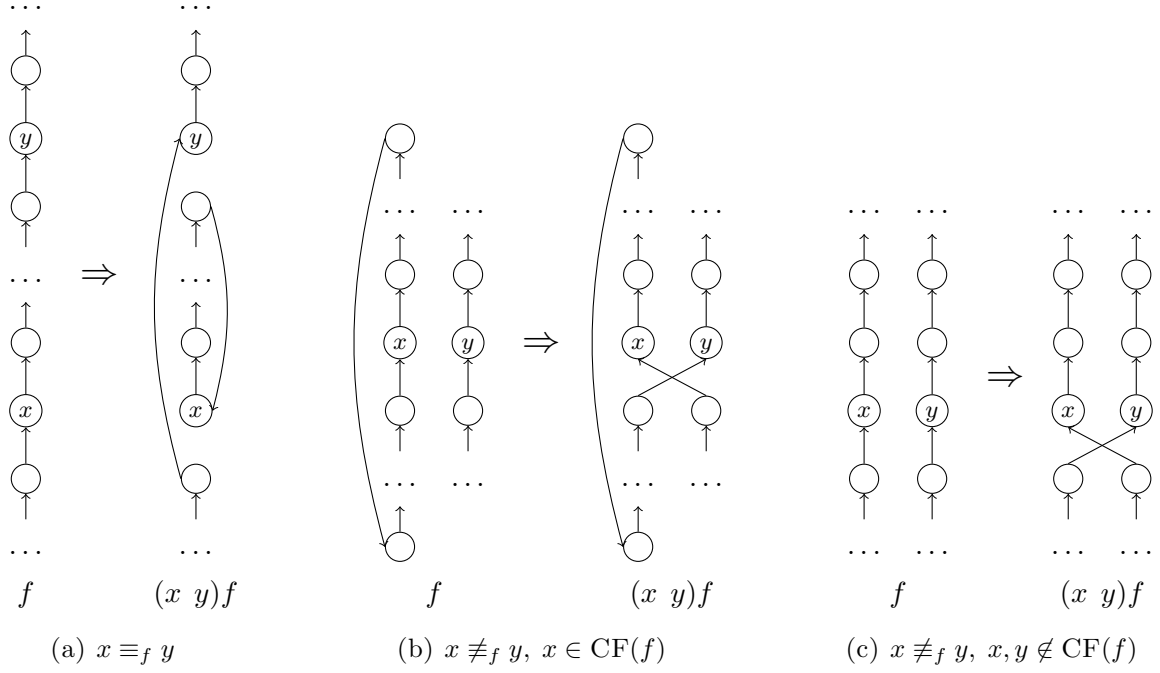


FIGURE 4. The functional digraphs of f and $(x \ y)f$ under different assumptions on x, y . The *functional digraph* of a recursive function f is a directed graph on vertices \mathbb{N} with an edge from x to y iff $f(x) = y$. Cycles of permutations are weakly connected components in the functional digraph. Barring exchange of x and y the shown cases (a)—(c) are exhaustive.

Applying this theorem yields that $\langle \text{Perm} \rangle = \mathcal{G}$. Since Perm is closed under inversion, it follows that every recursive permutation can be expressed as a finite product of members of Perm. By Lemma 10.1 there is a $k \in \mathcal{G} \setminus \text{Perm}$. Then there is a decomposition $k = f_1 \dots f_n$ with $f_i \in \text{Perm}$. Now an index $1 \leq i < n$ must exist such that $f_1 \dots f_i \in \text{Perm}$ and $(f_1 \dots f_i)f_{i+1} \notin \text{Perm}$. This proves

Corollary 11.1. Perm is not a group. More precisely, there are $f, g \in \text{Perm}$ such that $fg \notin \text{Perm}$. \square

The proof did not give examples of such f, g . They may be found by factoring $k \in \mathcal{G} \setminus \text{Perm}$ into members of Perm. Such a factorisation exists for every such k and always yields counterexamples to the closure of Perm, as seen above.

While Perm is not closed under multiplication with itself, one might expect that changes with finite support do not disturb cycle decidability. This is only partially true. Perm is closed under multiplication with F , the set of permutations with finite support, but a decider for such a product cannot always be computed.

Lemma 11.2. Let $f \in \text{Perm}$. There exists a recursive function $\rho\langle x, y \rangle$ such that for all x, y the function $\varphi_{\rho\langle x, y \rangle}$ decides the permutation $(x \ y)f$ iff $\text{CF}(f)$ is decidable. In this case ρ can be constructed from f , a decider for its cycles and a decider for its cycle finiteness problem.

Proof. We begin the proof with “ \Leftarrow ” which uses all cases depicted in Figure 4. This figure will be used as an argument in place of a formal calculation in the following proofs.

“ \Leftarrow ”: Suppose $\text{CF}(f)$ is decidable. We distinguish three cases (a)—(c) which correspond to the pictures in Figure 4. Which case applies is decidable by our prerequisites.

(a) If $x \equiv_f y$, then either $x = y$, which is trivial, or there is an $i > 0$ such that $f^i(x) = y$ or $f^i(y) = x$. By computing $\mu i[i > 0 \wedge \{f^i(x), f^i(y)\} \cap \{x, y\} \neq \emptyset]$, which terminates, we can

find i and determine which of x and y is the “upper” and which is the “lower” number in the functional digraph. We can assume, by symmetry, that x is the lower number, i.e. $f^i(x) = y$ with $i > 0$. In $(x \ y)f$ all cycles of f except $[x]_f = [y]_f$ are left untouched. The cycle $[x]_f$ is split into two cycles: one contains $x, f(x), \dots, f^{i-1}(x) = f^{-1}(y)$ and the other contains the rest. An algorithm to decide the cycles of $(x \ y)f$ is obvious.

(b) Assume $x \not\equiv_f y$ and $x \in \text{CF}(f) \vee y \in \text{CF}(f)$. Without loss of generality let $x \in \text{CF}(f)$. The cycles of f outside of $[x]_f, [y]_f$ are unchanged in $(x \ y)f$. The cycles $[x]_f$ and $[y]_f$ fuse in $(x \ y)f$ as can be seen in Figure 4b: take any two $i, j > 0$, then follow the arrows $f^{-i}(y), \dots, f^{-1}(y), x, f(x), \dots, f^{-1}(x), y, f(y), \dots, f^j(y)$.

(c) Assume finally $x \not\equiv_f y$ and $x \notin \text{CF}(f) \wedge y \notin \text{CF}(f)$. Once again the cycles of f apart from $[x]_f$ and $[y]_f$ remain unchanged. In $(x \ y)f$ the cycles $[x]_f$ and $[y]_f$ are split into four sets and recombined into two different cycles as Figure 4c indicates. More specifically: because $[x]_f$ is infinite, each $x' \in [x]_f$ has a unique number $k_x(x') = \xi k[f^k(x) = x']$; similarly for $y' \in [y]_f$. Those x' with $k_x(x') < 0$ are in $[y]_{(x \ y)f}$ and those with $k_x(x') \geq 0$ are in $[x]_{(x \ y)f}$. Analogously $y' \in [y]_f$ is $y' \in [y]_{(x \ y)f} \Leftrightarrow k_y(y') \geq 0$ and in $[x]_{(x \ y)f}$ else.

Given f and deciders for the cycles of f and its cycle finiteness problem, the construction of a decider for $(x \ y)f$ is uniform in x and y .

“ \Rightarrow ”: This direction is not uniform in f . If f has no infinite cycles, then a decider for $\text{CF}(f)$ is constant and therefore computable. If f has at least one infinite cycle, let y_0 denote a number in an infinite cycle of f . We want to decide $\text{CF}(f)$. Let x be given. If $x \equiv_f y_0$, then x is obviously in an infinite cycle. Else we obtain a decider for $(x \ y_0)f$. From Figure 4, with $x \not\equiv_f y_0$, we see that

$$x \in \text{CF}(f) \vee y_0 \in \text{CF}(f) \Leftrightarrow x \equiv_{(x \ y_0)f} y_0$$

whose RHS is decidable and whose LHS is equivalent to $x \in \text{CF}(f)$, by the choice of y_0 . This gives a method to decide cycle finiteness. \square

Lemma 11.3. There is an algorithm uniform in $f \in \text{Perm}_{\text{CF}}$, a decider for its cycles and its cycle finiteness problem which computes to every pair x, y a decider for the cycles and the cycle finiteness problem of $(x \ y)f$.

Proof. Lemma 11.2 yields a decider for the cycles of $(x \ y)f$. Using Figure 4, a decider for the cycles of f and for its cycle finiteness problem, it is easy to describe a decider for the cycle finiteness problem of $(x \ y)f$. \square

The transpositions can be enumerated via $\langle x, y \rangle \mapsto (x \ y)$, for which an algorithm τ is immediate. The term “transposition” is meant to include the improper transposition id , as it is also enumerated by τ . The finitary permutations F can be enumerated via the following algorithm which uses the fact that F is the subgroup generated by transpositions: given any number z , decode it into $z = \langle n, z' \rangle$ and then decode $z' = \langle x_1, x_2, \dots, x_n \rangle$. This tuple can be mapped uniformly effectively to a program for computing $\varphi_{\tau(x_1)} \dots \varphi_{\tau(x_n)}$. Call this enumeration η .

Theorem 11.4. For $f \in \text{Perm}$ and $a, b \in F$ it is $afb \in \text{Perm}$. If $f \in \text{Perm}_{\text{CF}}$ then $afb \in \text{Perm}_{\text{CF}}$. Given $f \in \text{Perm}_{\text{CF}}$, a decider for its cycles and its cycle finiteness problem, as well η indices for a and b , we can find a decider for the cycles and the cycle finiteness problem of afb .

Proof. We first show the non-constructive part with $f \in \text{Perm}$. It suffices to prove the statement under the assumption that $b = \text{id}$: assume we have shown $\forall f \in \text{Perm} \forall a \in F : af \in \text{Perm}$, then $afb \in \text{Perm}$ iff $fb \in \text{Perm}$ iff $b^{-1}f^{-1} \in \text{Perm}$ which is true because $b^{-1} \in F$ and Perm is closed under inversion. Since a has finite support, we only need to consider the case where $a = (x \ y)$ is a transposition and can proceed by induction. Then the proof is almost taken care of in Lemma 11.2. Indeed reading the “ \Leftarrow ” proof with an oracle for $\text{CF}(f)$ (instead of the assumption that this problem is decidable) gives a decider for $(x \ y)f$ in each of the three cases considered there. The oracle is only used to decide which case applied; the constructed deciders are recursive in each separate case in f and a decider for its cycles only. Thus a recursive decider always exists and $(x \ y)f \in \text{Perm}$.

The constructive part with $f \in \text{Perm}_{\text{CF}}$ works in the same way but must be executed more carefully. Let $a = \varphi_{\eta(z)}$, $b = \varphi_{\eta(w)}$. The η index z encodes a decomposition of a into transpositions. Using Lemma 11.3 inductively on this decomposition, we obtain deciders for af as follows. At the beginning of each induction step we have a permutation g (initially $g = f$), deciders for its cycles and cycle finiteness problem, and the two constituents x, y of the transposition $(x \ y)$. Applying Lemma 11.3 we obtain a decider for the cycles and the cycle finiteness problem of $(x \ y)g$. The next induction step can then be performed on $(x \ y)g$.

After this inductive process, we have a decider for the cycles and the cycle finiteness problem of af , which also decide these problems for $f' = f^{-1}a^{-1}$. From w , compute w' such that $\varphi_{\eta(w')} = b^{-1} \in F$ and apply the same procedure to $b^{-1}f'$. The deciders we obtain at the end decide the cycles and the cycle finiteness problem of $b^{-1}f' = b^{-1}f^{-1}a^{-1}$ and also those of its inverse, afb . \square

Corollary 11.5.

- (i) Perm is closed under multiplication with finitary permutations.
- (ii) Perm_{CF} is constructively closed under multiplication with finitary permutations.
- (iii) There exists a $g \in \text{Perm}$ such that there is no recursive mapping of pairs $\langle x, y \rangle$ to a decider for the permutation $(x \ y)g$.

The term “constructively closed” in the second part of the theorem means that if $f \in \text{Perm}_{\text{CF}}$ and $a, b \in F$, witnesses for the statement $afb \in \text{Perm}_{\text{CF}}$, i.e. the two deciders, can be computed uniformly in f , η indices for a and b , and witnesses for $f \in \text{Perm}_{\text{CF}}$. The third part of the corollary then means that Perm is not constructively closed: agb is in Perm, but a witness for this statement cannot be computed uniformly in η indices for a and b , even when g and a decider for its cycles are fixed.

Proof. (i) and (ii) are Theorem 11.4; (iii) is Lemma 11.2 applied to the permutation from Lemma 8.2. \square

REFERENCES

- [BMMN98] Meenaxi Bhattacharjee, Dugald Macpherson, Rögnvaldur G. Möller, and Peter M. Neumann. *Notes on Infinite Permutation Groups*. Number 1698 in Lecture Notes in Mathematics. Springer, 1998.
- [Dav58] Martin Davis. *Computability & Unsolvability*. Dover Books on Computer Science Series. Dover, 1958.
- [DM96] John D. Dixon and Brian Mortimer. *Permutation Groups*. Graduate Texts in Mathematics. Springer New York, 1996.
- [Hig90] Graham Higman. Transversals and conjugacy in the group of recursive permutations. In L. G. Kovács, editor, *Groups—Canberra 1989: Australian National University Group Theory Program*, pages 142–160. Springer Berlin Heidelberg, 1990.
- [Ken62] Clement F. Kent. Constructive analogues of the group of permutations of the natural numbers. *Transactions of the American Mathematical Society*, 104(2):347–362, 1962.
- [Leh09] Eero Lehtonen. An undecidable permutation of the natural numbers. In Olivier Bournez and Igor Potapov, editors, *Reachability Problems*, volume 5797 of *Lecture Notes in Computer Science*, pages 120–126. Springer Berlin Heidelberg, 2009.
- [LP98] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 1998.
- [Myh59] John Myhill. Recursive digraphs, splinters and cylinders. *Mathematische Annalen*, 138(3):211–218, 1959.
- [Rog58] Hartley Rogers, Jr. Gödel numberings of partial recursive functions. *The Journal of Symbolic Logic*, 23(3):pp. 331–341, 1958.
- [Rog87] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987.
- [Sch16] Bernd Schröder. *Ordered Sets*. Birkhäuser Basel, second edition, 2016.
- [vL15] Jan van Leeuwen. A note on recursively enumerable classes of partial recursive functions. In *Technical Report Series, issue UU-CS-2015-001*. UU BETA ICS Departement Informatica, 2015.

OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG, MAGDEBURG, GERMANY
E-mail address: `tboege@st.ovgu.de`